Fast and Accurate Triangle Counting in Graph Streams Using Predictions

Cristian Boldrin Fabio Vandin

University of Padova, Italy

cristian.boldrin.2@phd.unipd.it

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

ICDM 2024

Abu Dhabi
December, 2024

Problem Definition

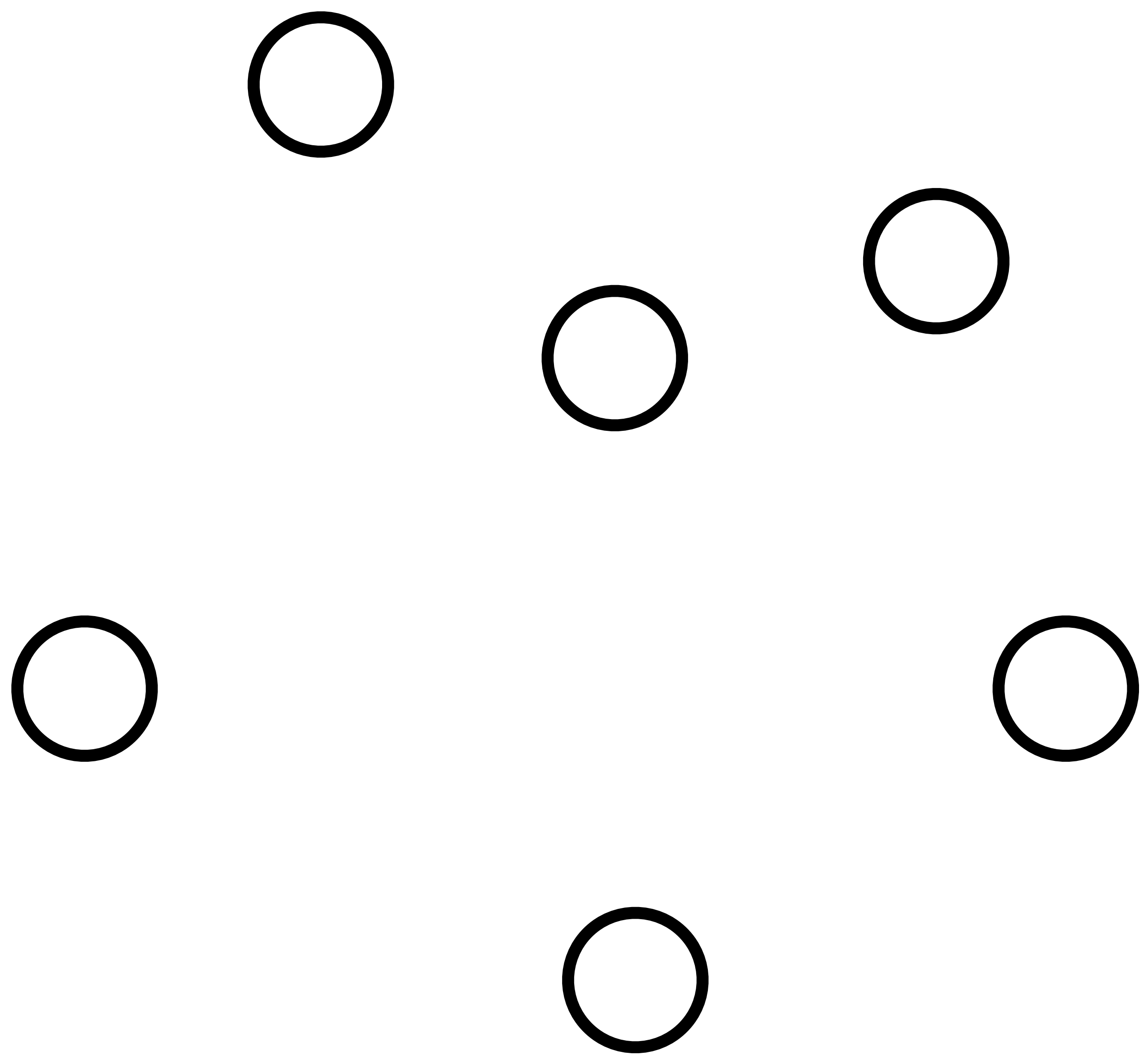
Problem: count the number of triangles in graphs.

Problem Definition

Problem: count the number of triangles in graphs.

Given:

- a set V of nodes, $\left|V\right| = n$

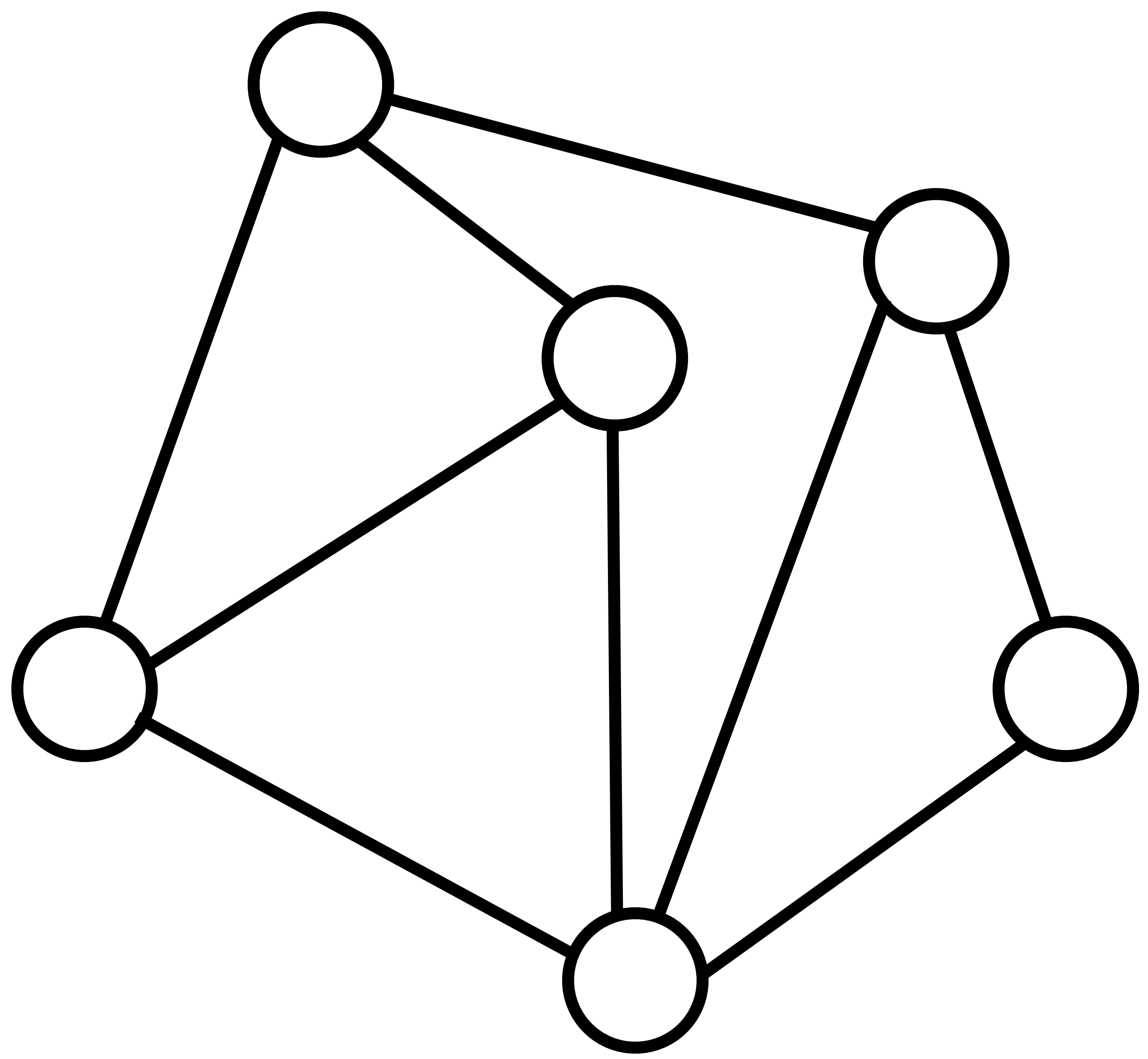


Problem Definition

Problem: count the number of triangles in graphs.

Given:

- a set V of nodes, $\left|V\right| = n$
- A set E of edges, $\left|E\right| = m$

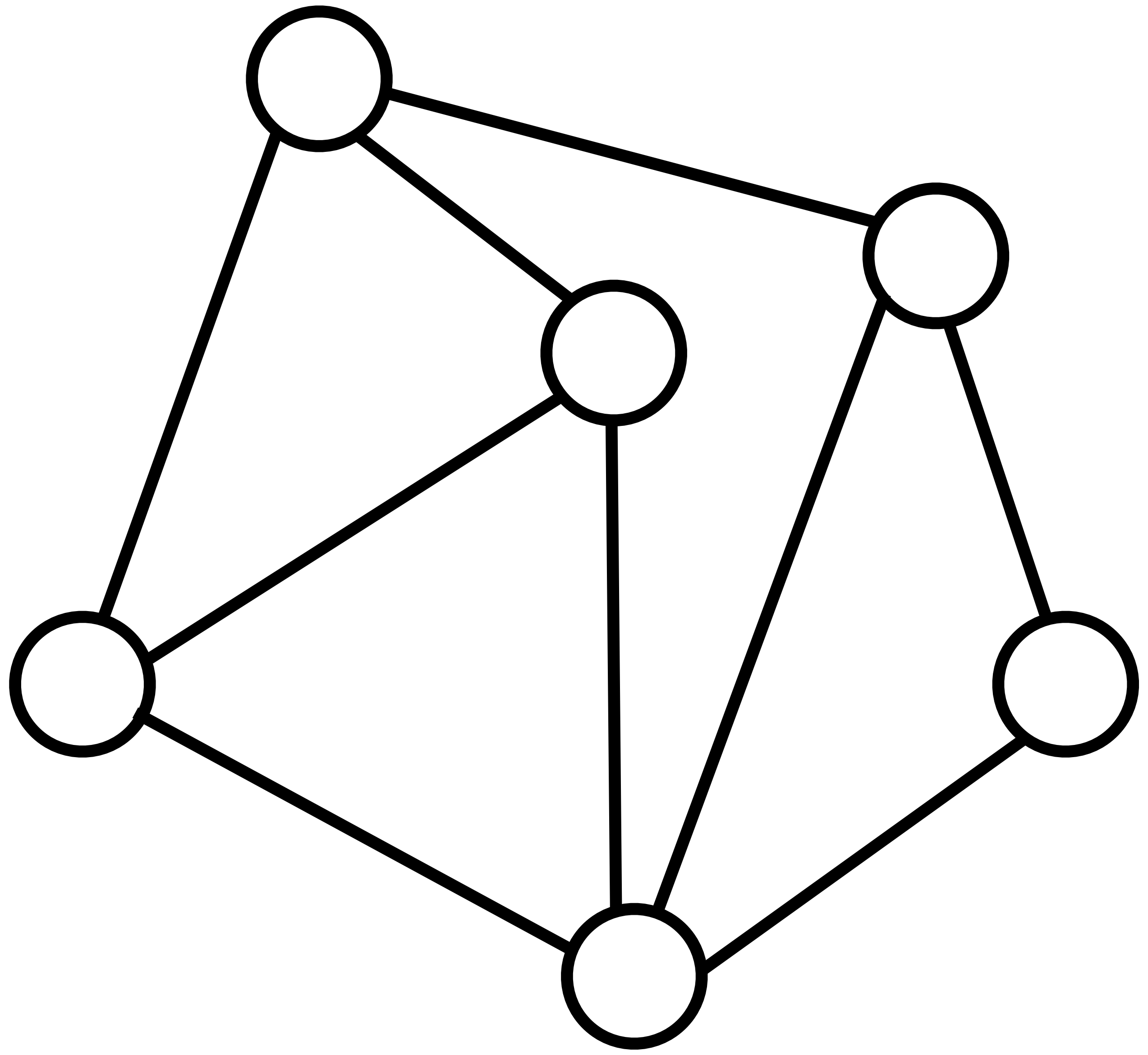


Problem Definition

Problem: count the number of triangles in graphs.

Given:

- a set V of nodes, $\left|V\right| = n$
- A set E of edges, $\left|E\right| = m$



Goal:

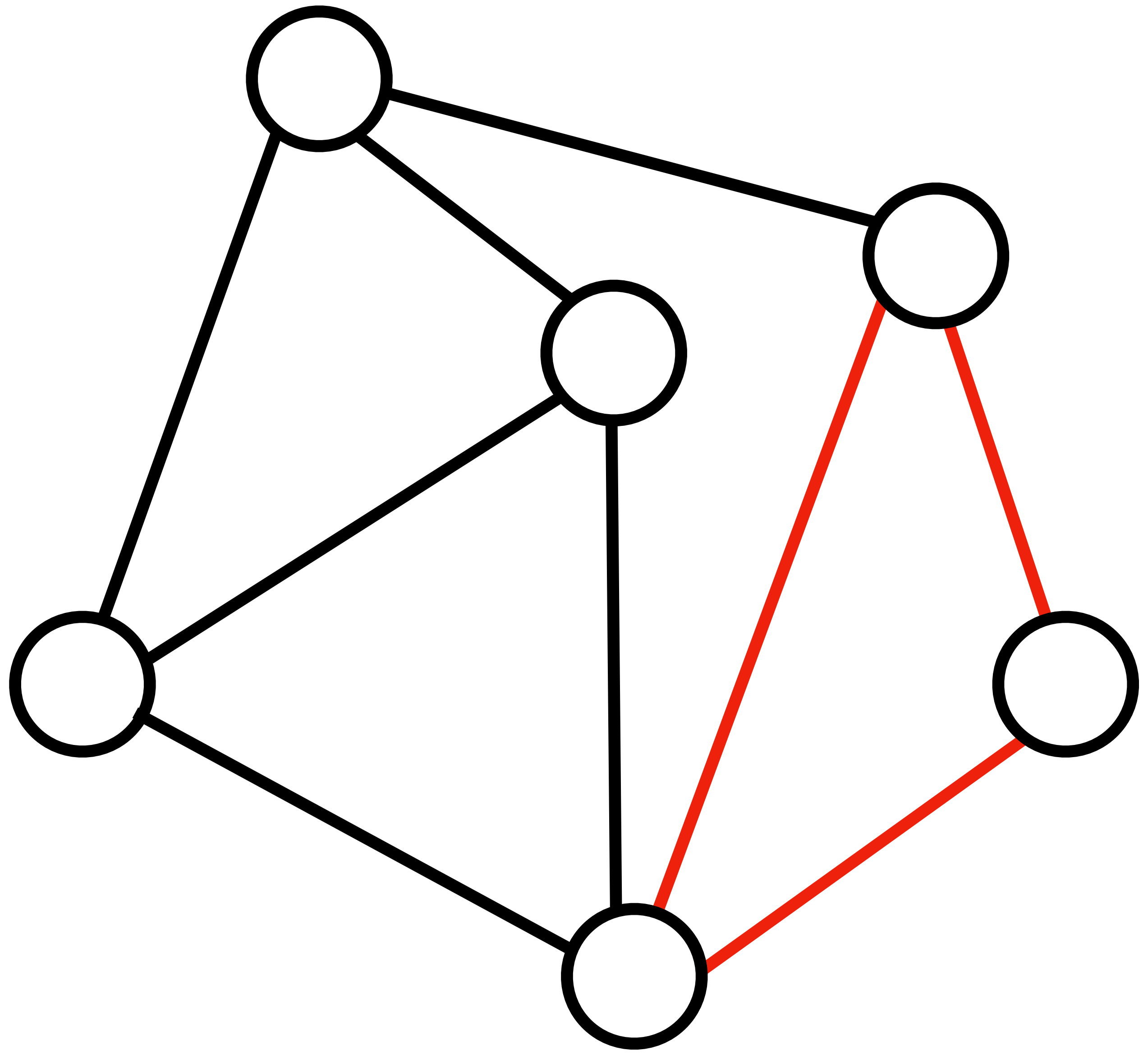
- Count the **global** number of triangles $\Delta = \{u, v, w\}$, where $\{u, v\}$, $\{w, u\}$, and $\{v, w\}$ are all in the set E of the edges

Problem Definition

Problem: count the number of triangles in graphs.

Given:

- a set V of nodes, $\left|V\right| = n$
- A set E of edges, $\left|E\right| = m$



Goal:

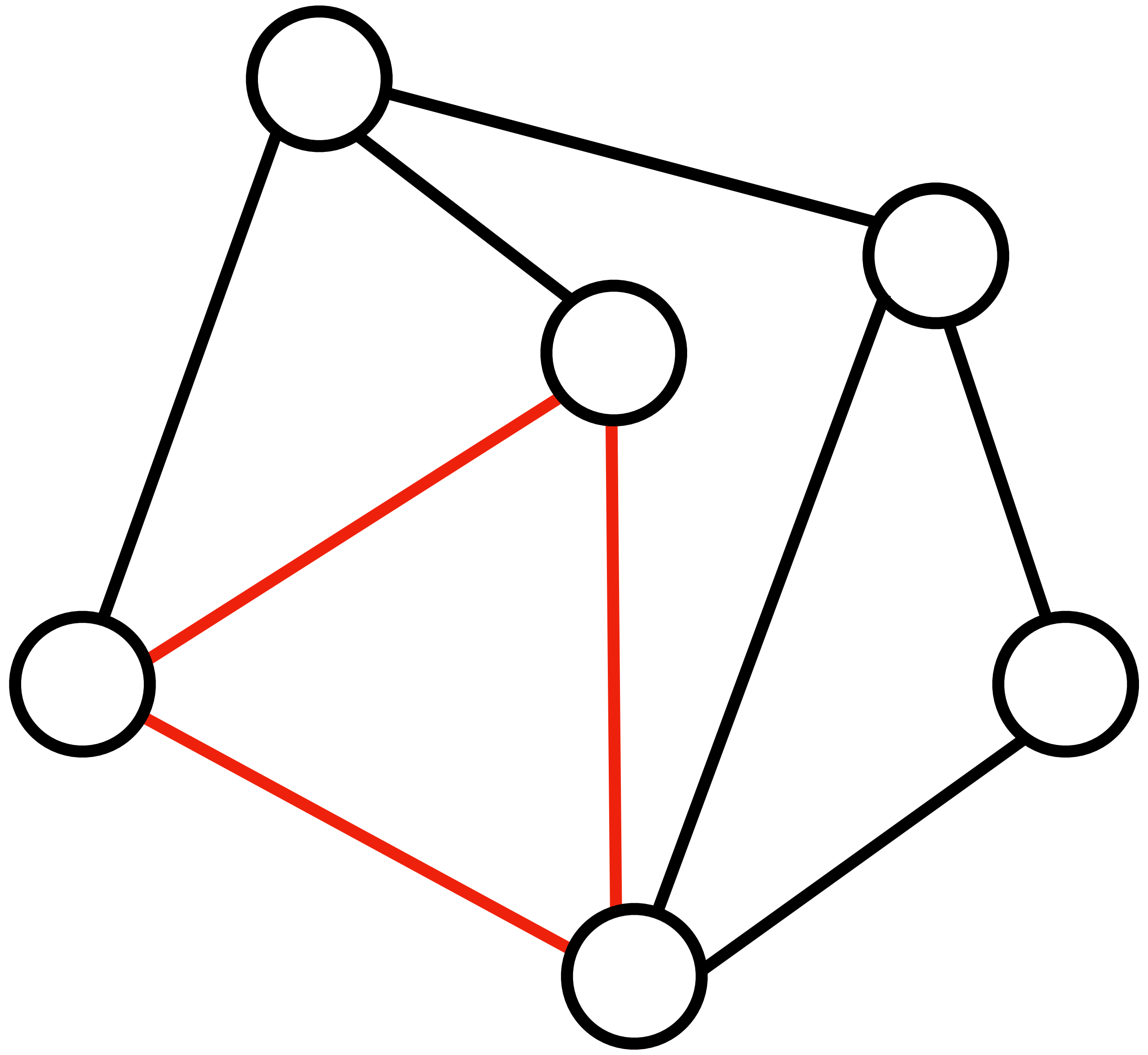
- Count the **global** number of triangles $\Delta = \{u, v, w\}$, where $\{u, v\}$, $\{w, u\}$, and $\{v, w\}$ are all in the set E of the edges

Problem Definition

Problem: count the number of triangles in graphs.

Given:

- a set V of nodes, $\left|V\right| = n$
- A set E of edges, $\left|E\right| = m$



Goal:

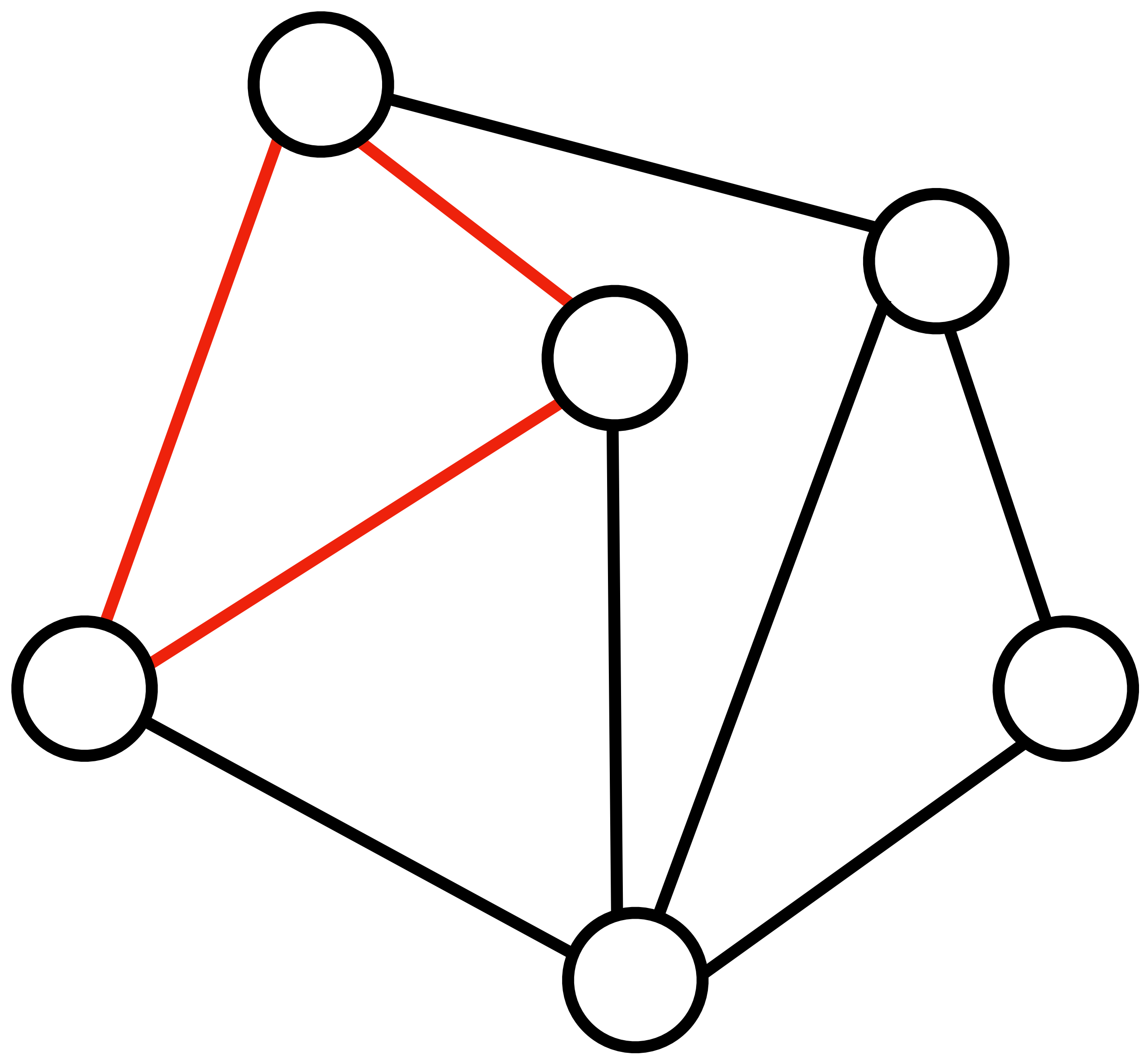
- Count the **global** number of triangles $\Delta = \{u, v, w\}$, where $\{u, v\}$, $\{w, u\}$, and $\{v, w\}$ are all in the set E of the edges

Problem Definition

Problem: count the number of triangles in graphs.

Given:

- a set V of nodes, $\left|V\right| = n$
- A set E of edges, $\left|E\right| = m$



Goal:

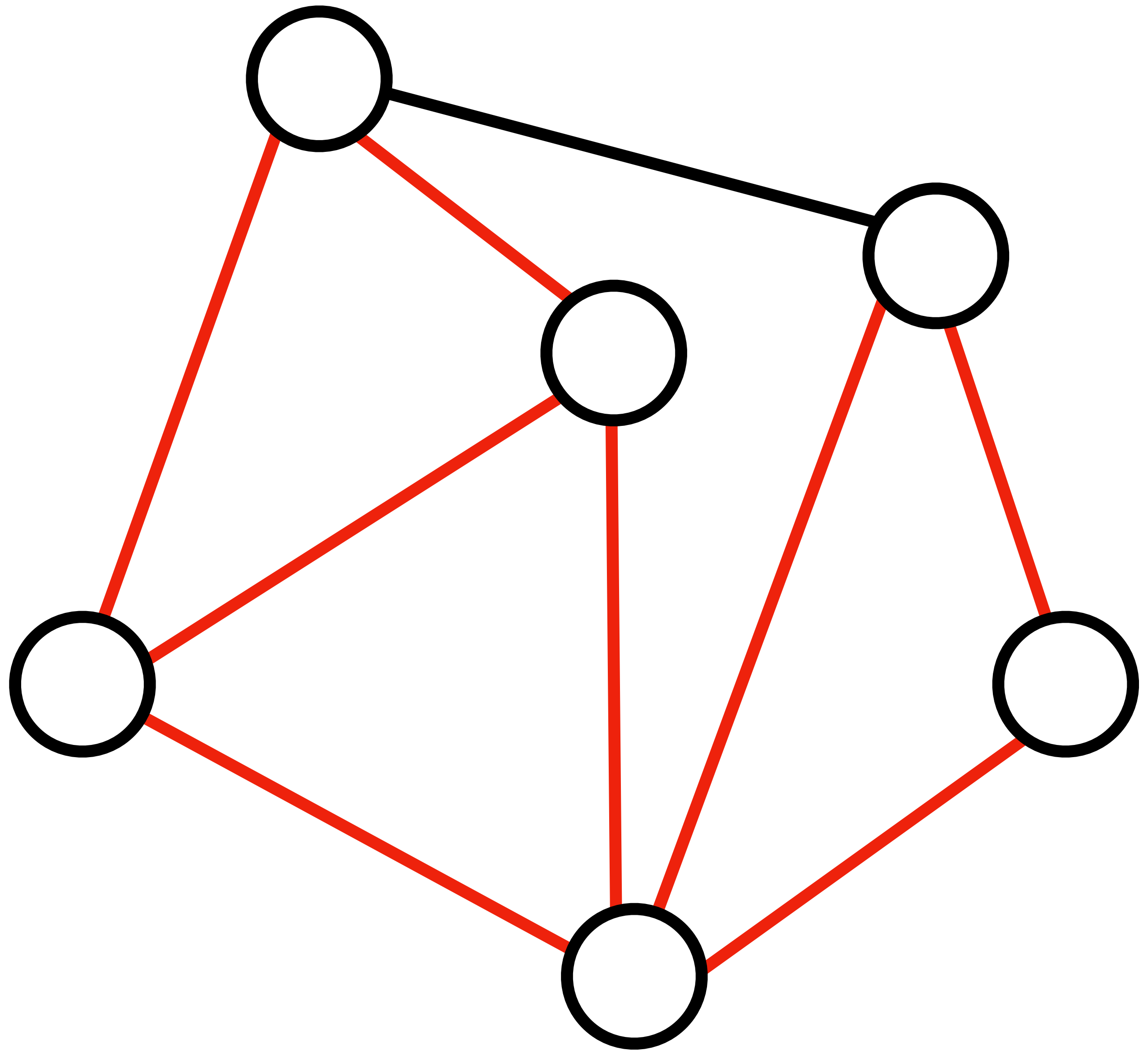
- Count the **global** number of triangles $\Delta = \{u, v, w\}$, where $\{u, v\}$, $\{w, u\}$, and $\{v, w\}$ are all in the set E of the edges

Problem Definition

Problem: count the number of triangles in graphs.

Given:

- a set V of nodes, $|V| = n$
- A set E of edges, $|E| = m$



Goal:

- Count the **global** number of triangles $\Delta = \{u, v, w\}$, where $\{u, v\}$, $\{w, u\}$, and $\{v, w\}$ are all in the set E of the edges

Applications:

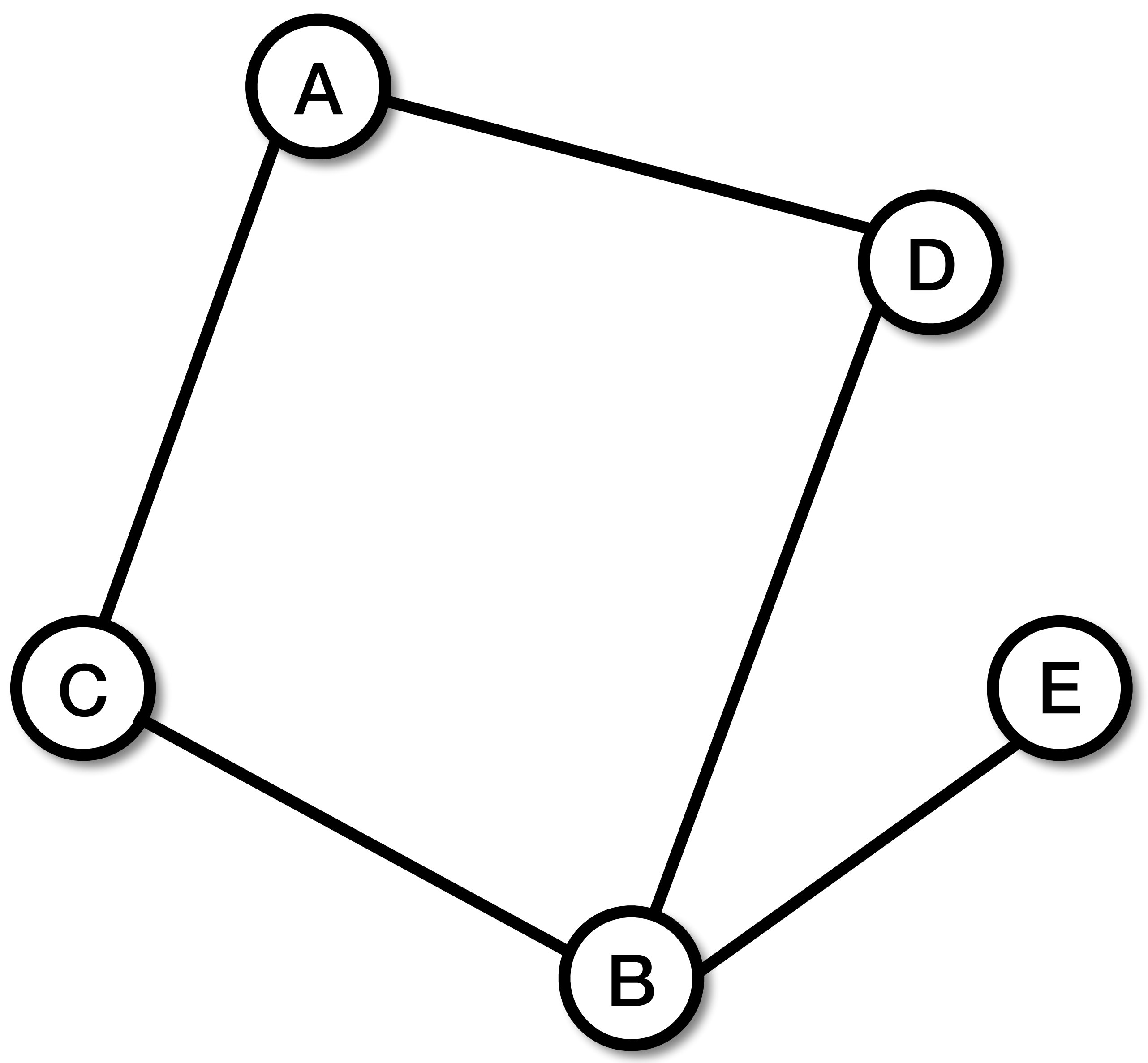
- Community detection
- Anomaly detection
- Molecular biology

Settings of our problem

Streaming model:

Edges are observed as a stream of updates in arbitrary order.

Updates: insertions and deletions.



Graph

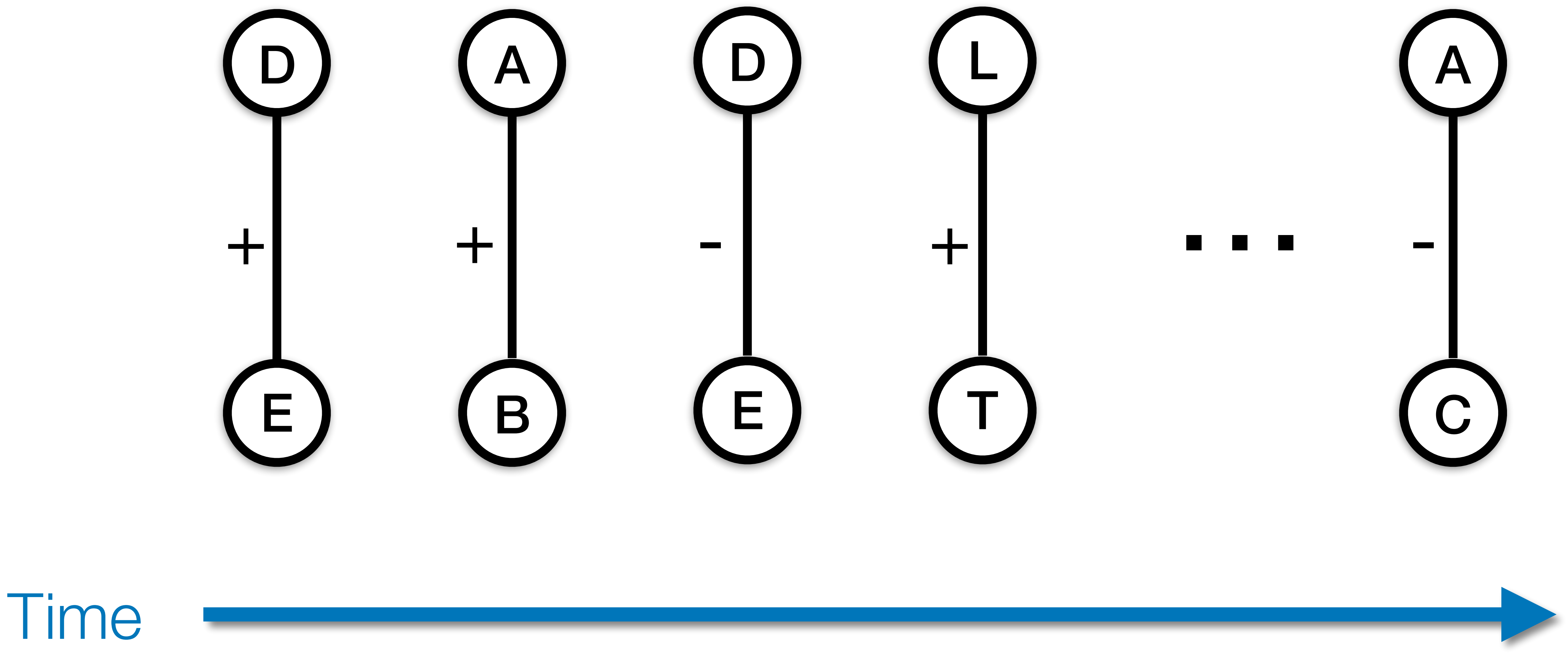
Settings of our problem

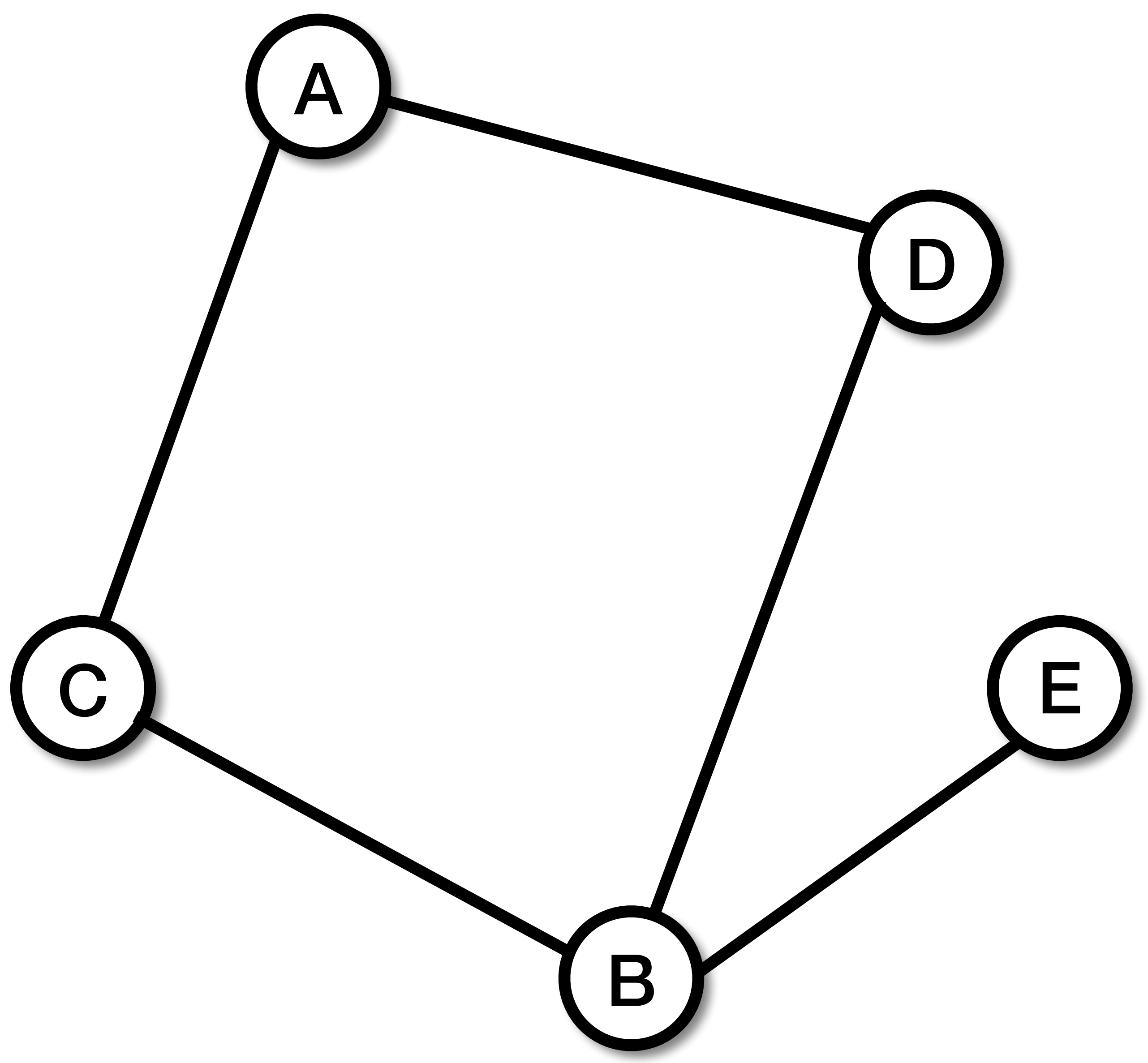
Streaming model:

Edges are observed as a stream of updates in arbitrary order.

Updates: insertions and deletions.

Stream Σ





Graph

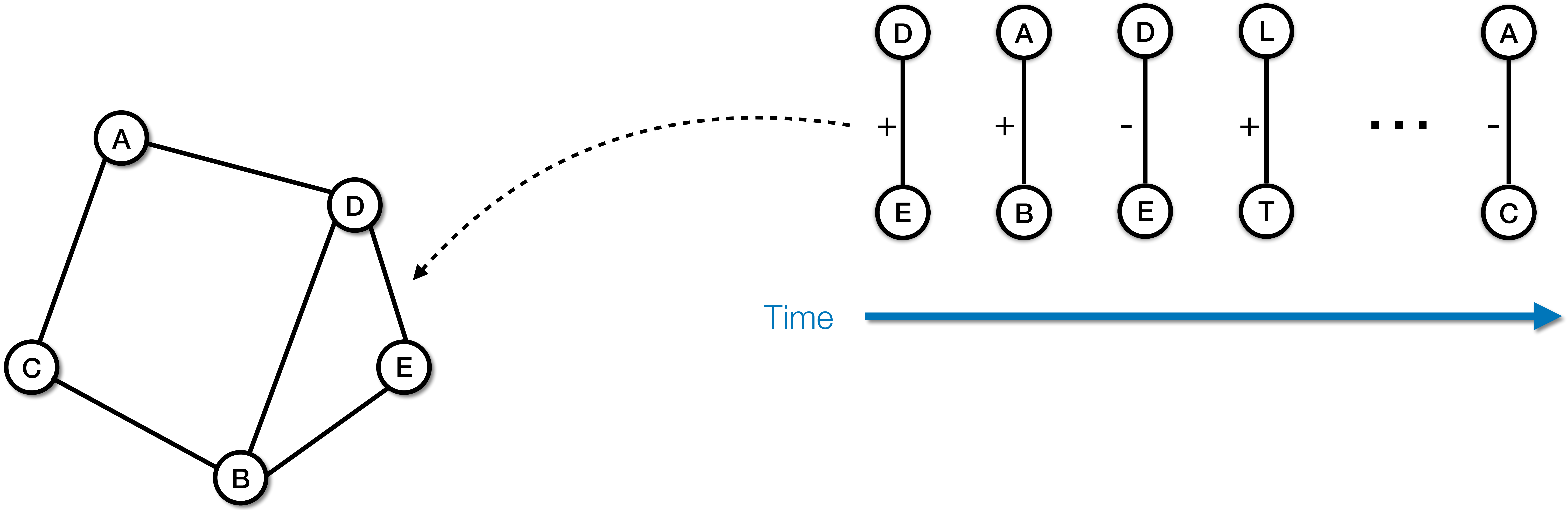
Settings of our problem

Streaming model.

Edges are observed as a stream of updates in arbitrary order.

Updates: insertions and deletions.

Stream Σ



Graph

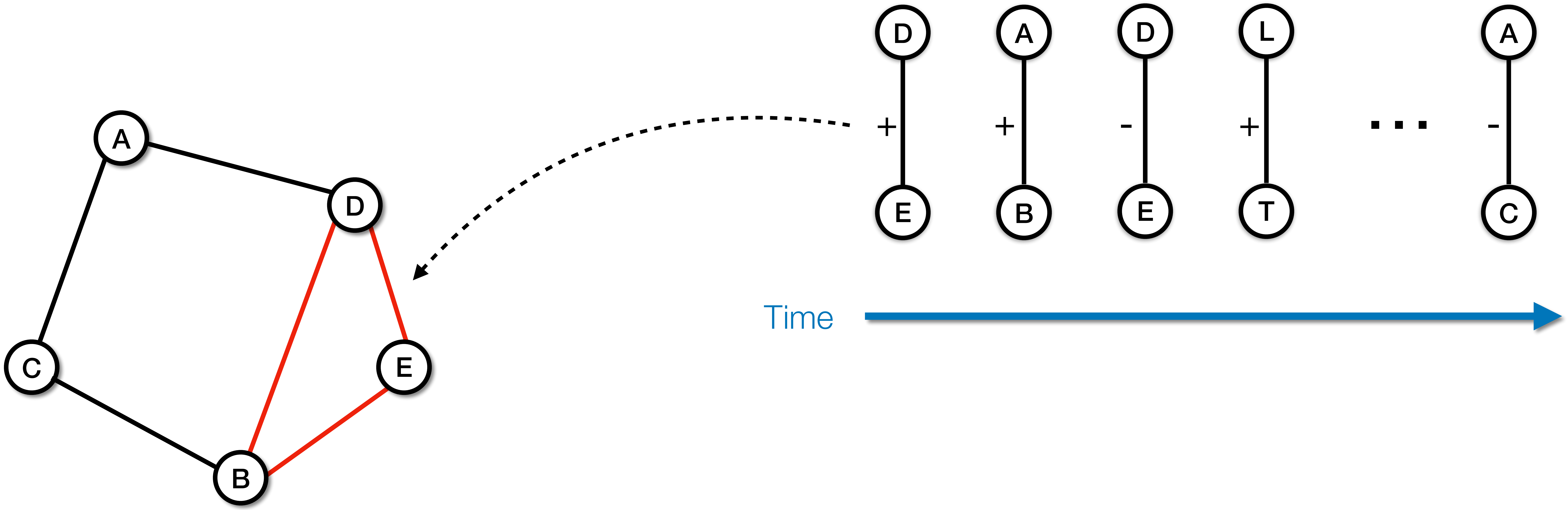
Settings of our problem

Streaming model:

Edges are observed as a stream of updates in arbitrary order.

Updates: insertions and deletions.

Stream Σ



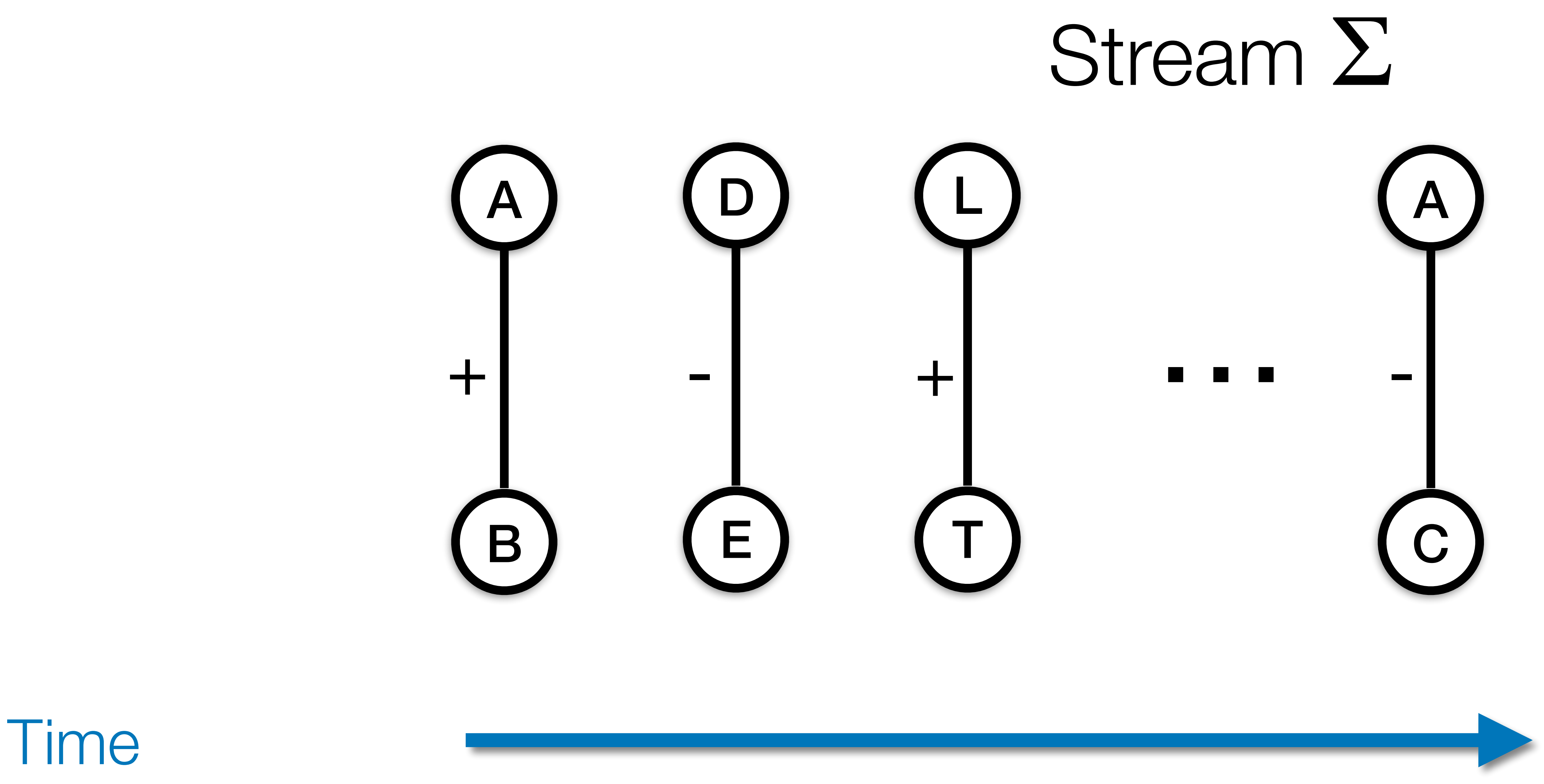
Graph

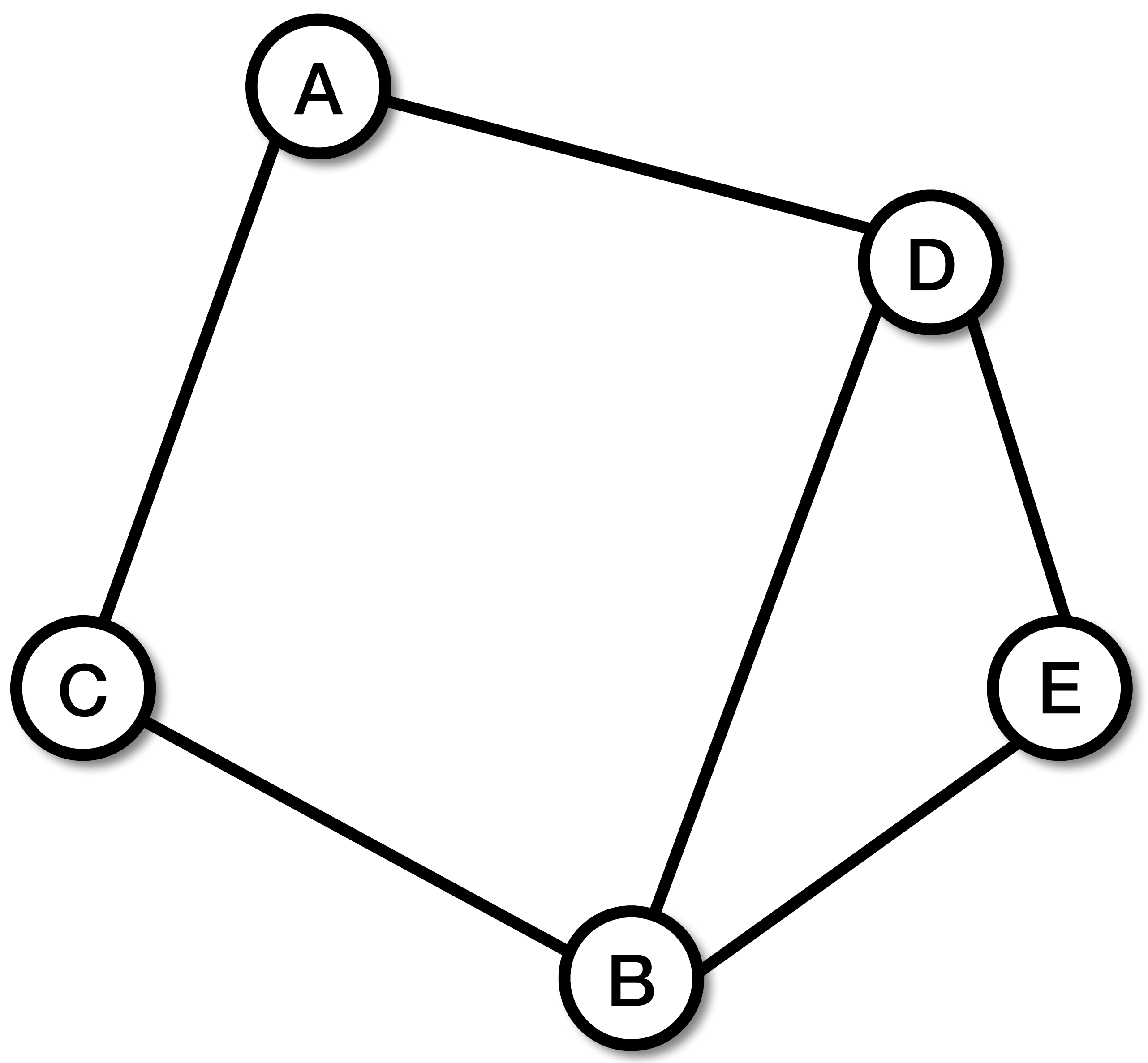
Settings of our problem

Streaming model:

Edges are observed as a stream of updates in arbitrary order.

Updates: insertions and deletions.





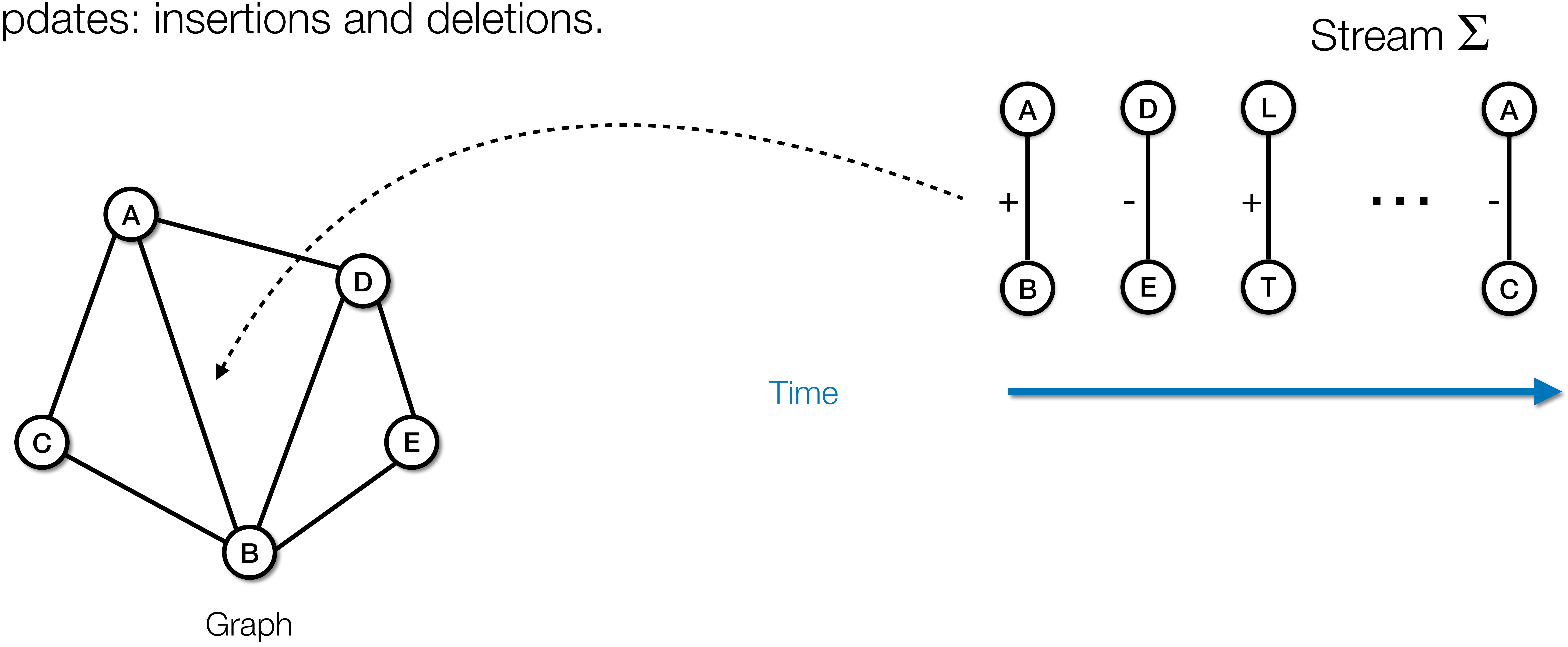
Graph

Settings of our problem

Streaming model:

Edges are observed as a stream of updates in arbitrary order.

Updates: insertions and deletions.

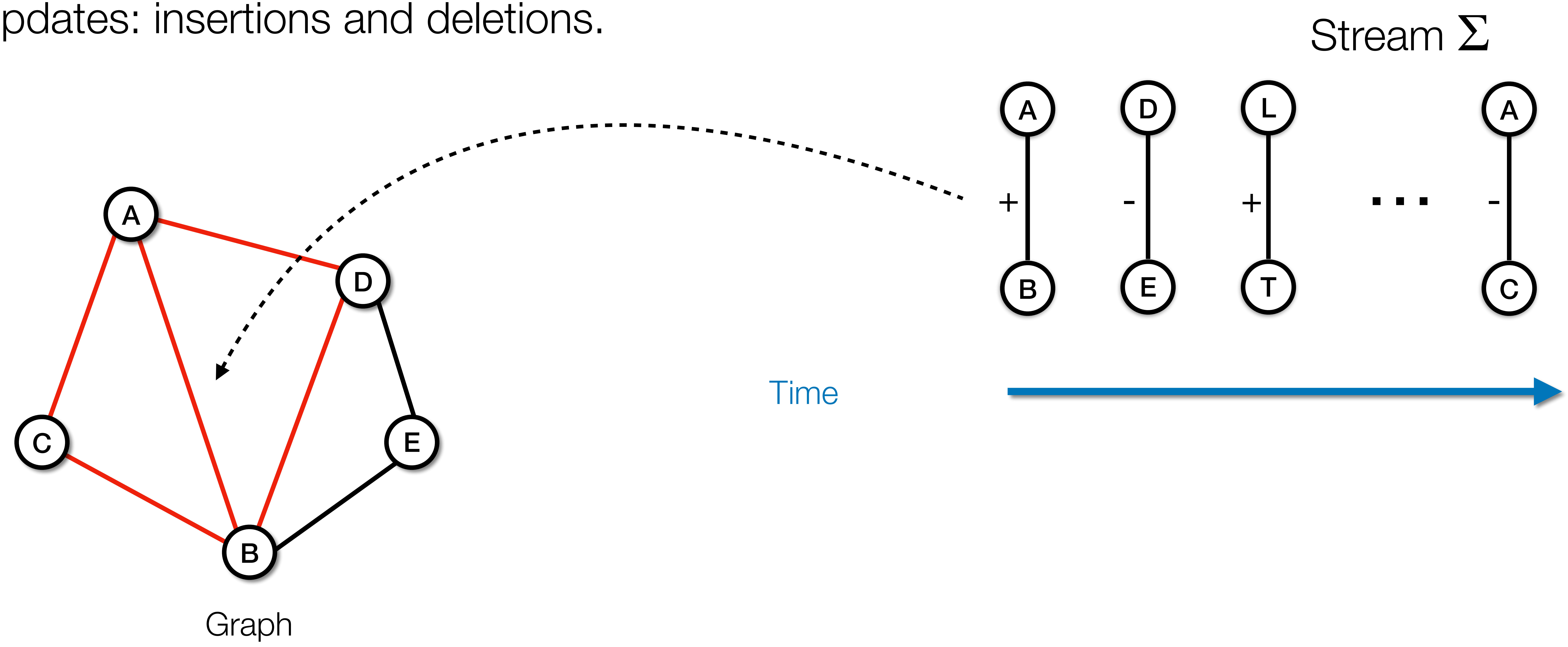


Settings of our problem

Streaming model:

Edges are observed as a stream of updates in arbitrary order.

Updates: insertions and deletions.

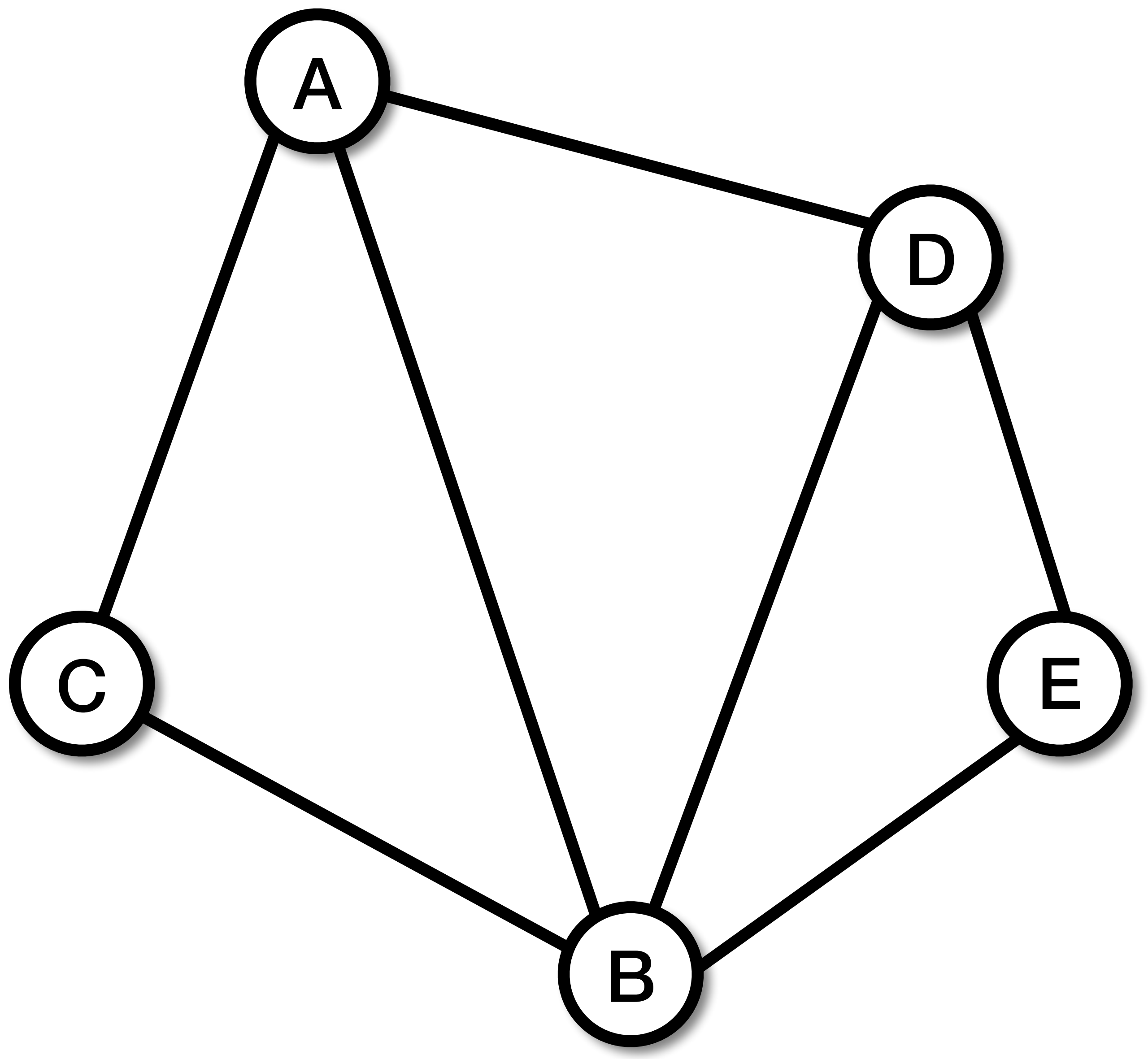


Settings of our problem

Streaming model:

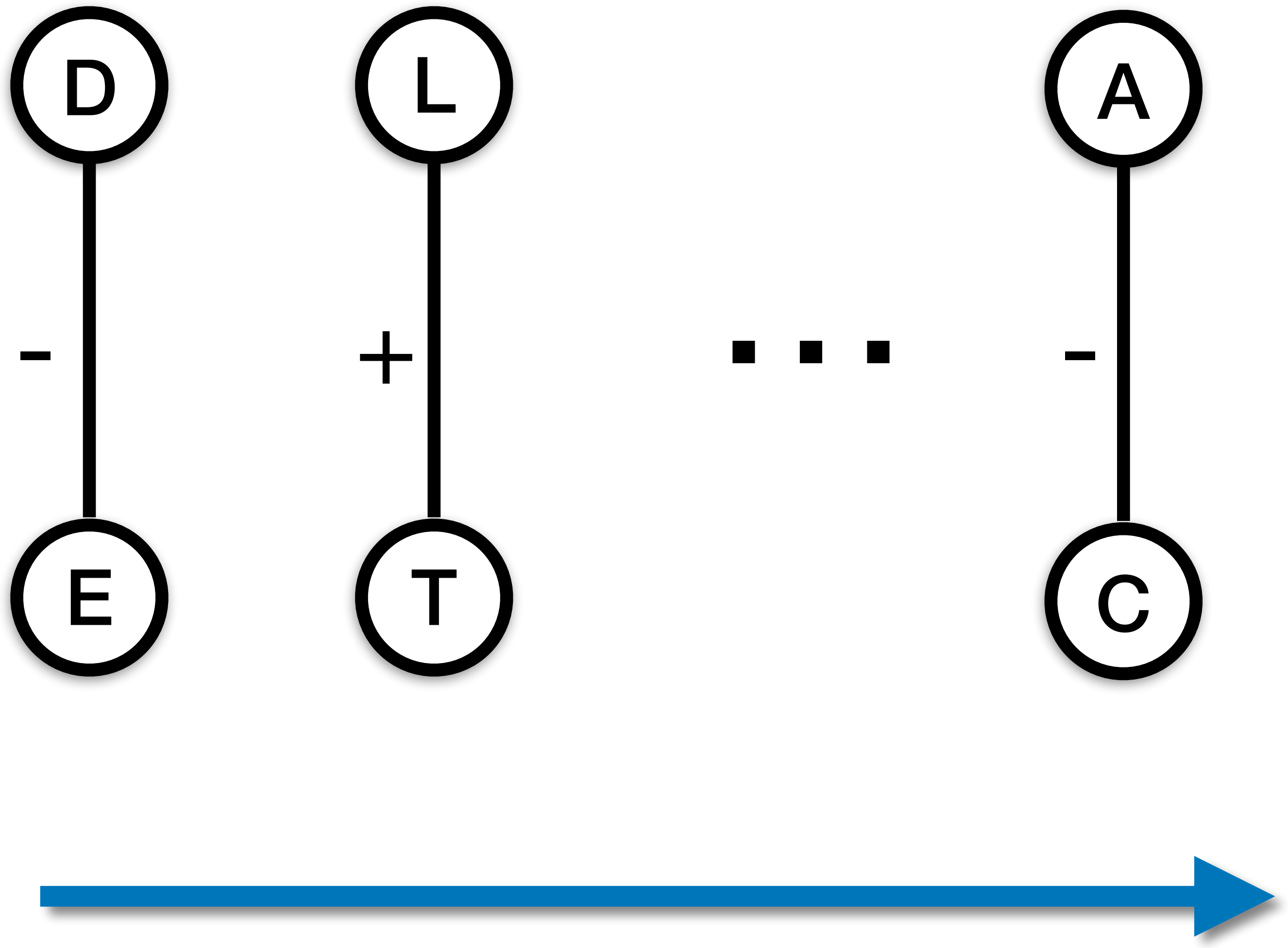
Edges are observed as a stream of updates in arbitrary order.

Updates: insertions and deletions.



Graph

Stream Σ



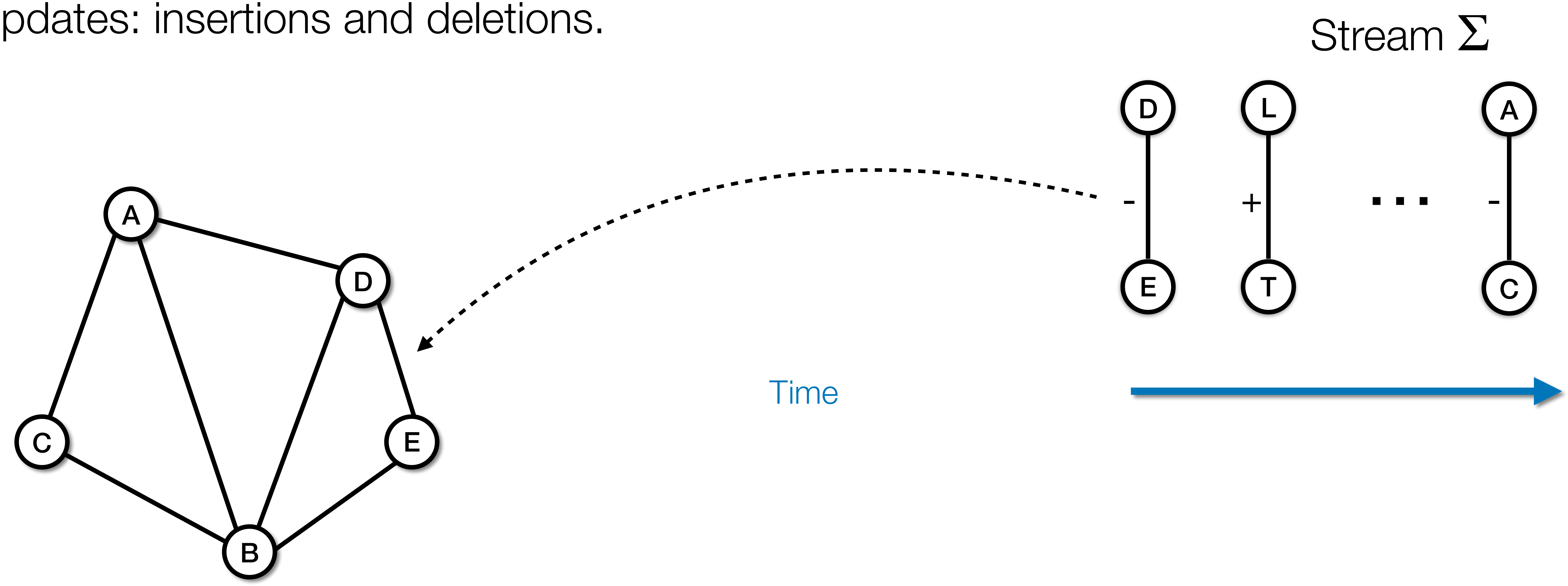
Time

Settings of our problem

Streaming model:

Edges are observed as a stream of updates in arbitrary order.

Updates: insertions and deletions.



Graph

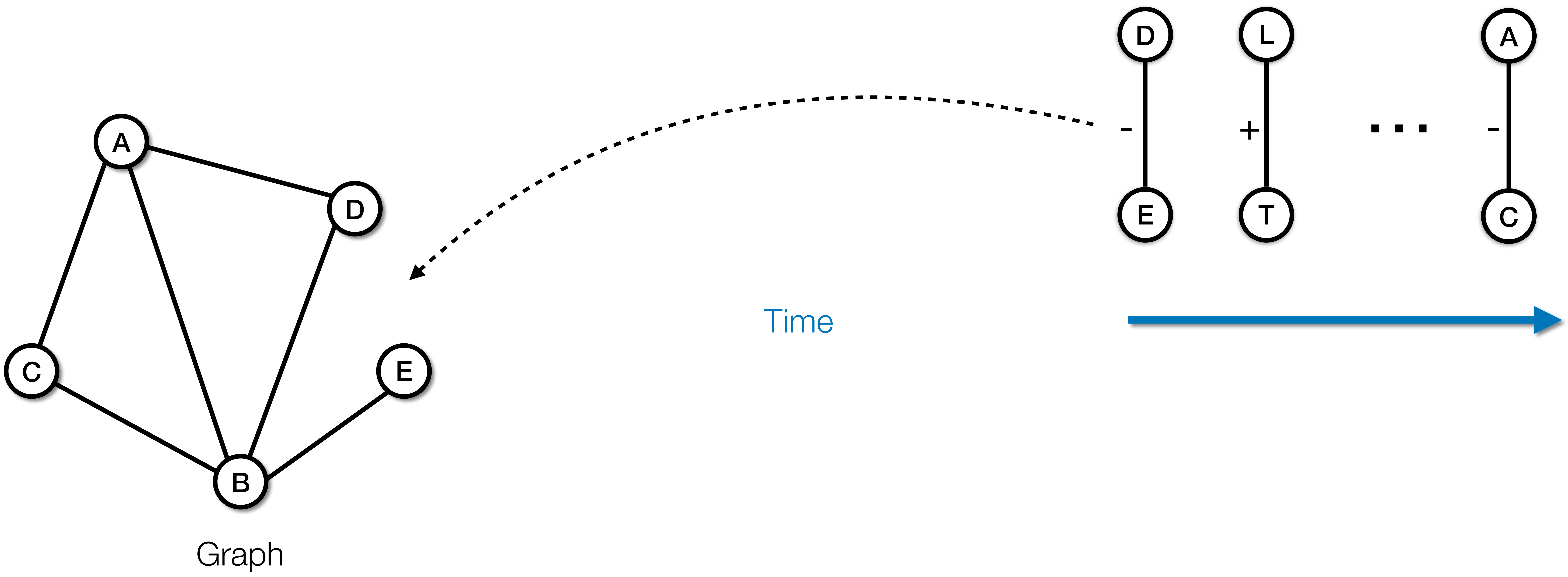
Settings of our problem

Streaming model:

Edges are observed as a stream of updates in arbitrary order.

Updates: insertions and deletions.

Stream Σ



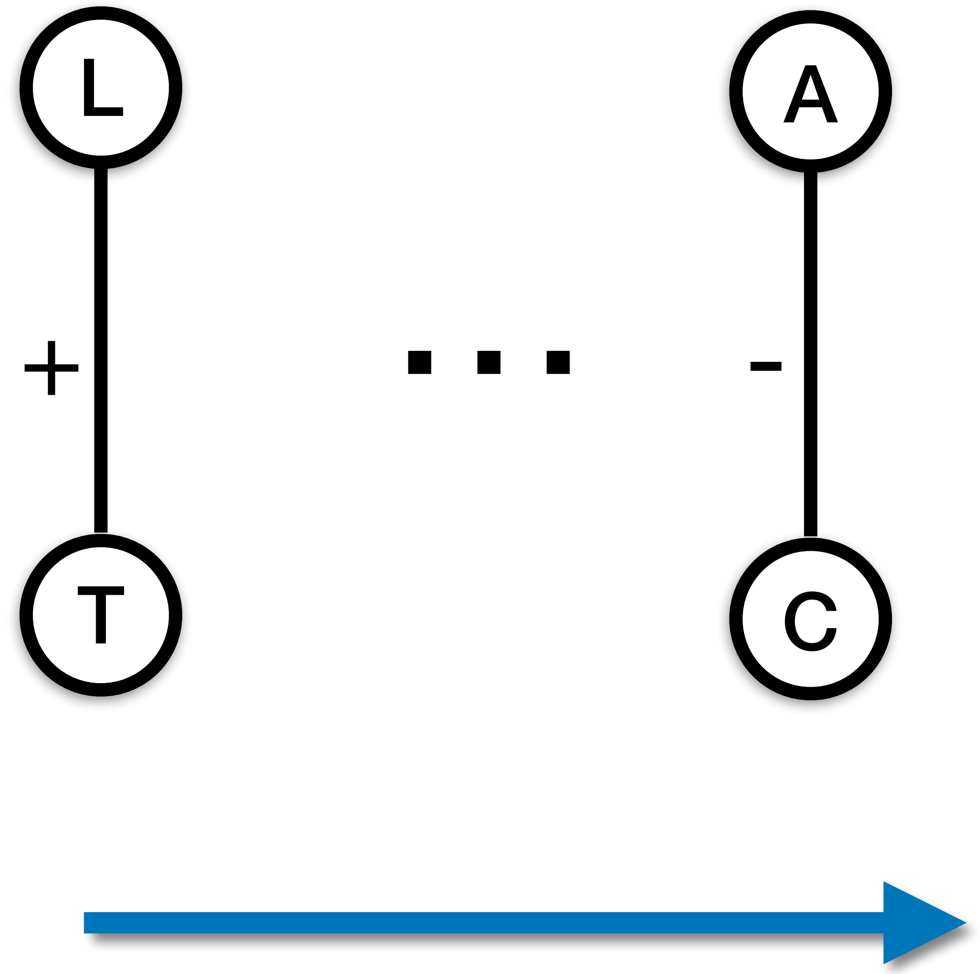
Settings of our problem

Streaming model:

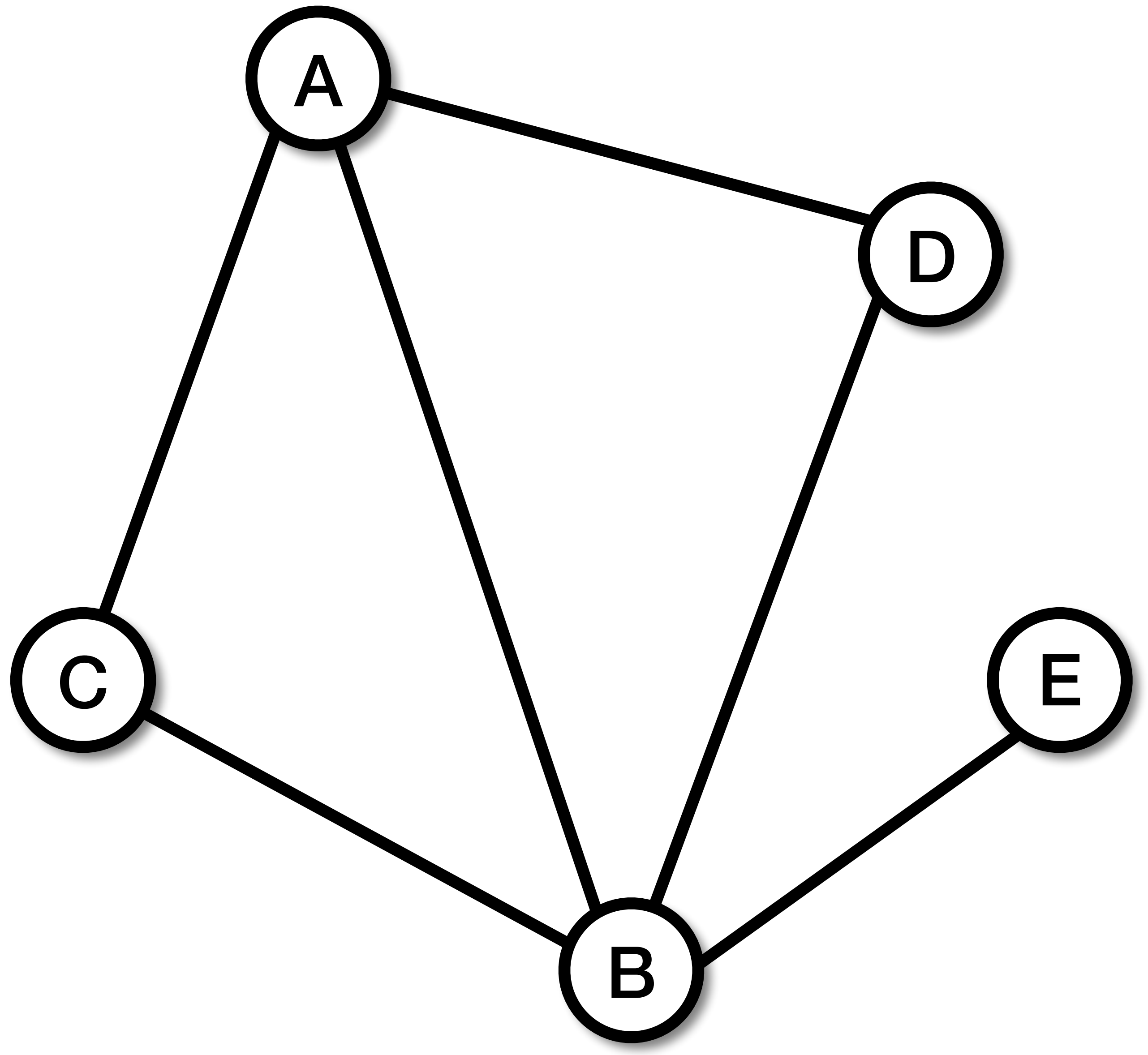
Edges are observed as a stream of updates in arbitrary order.

Updates: insertions and deletions.

Stream Σ



Time



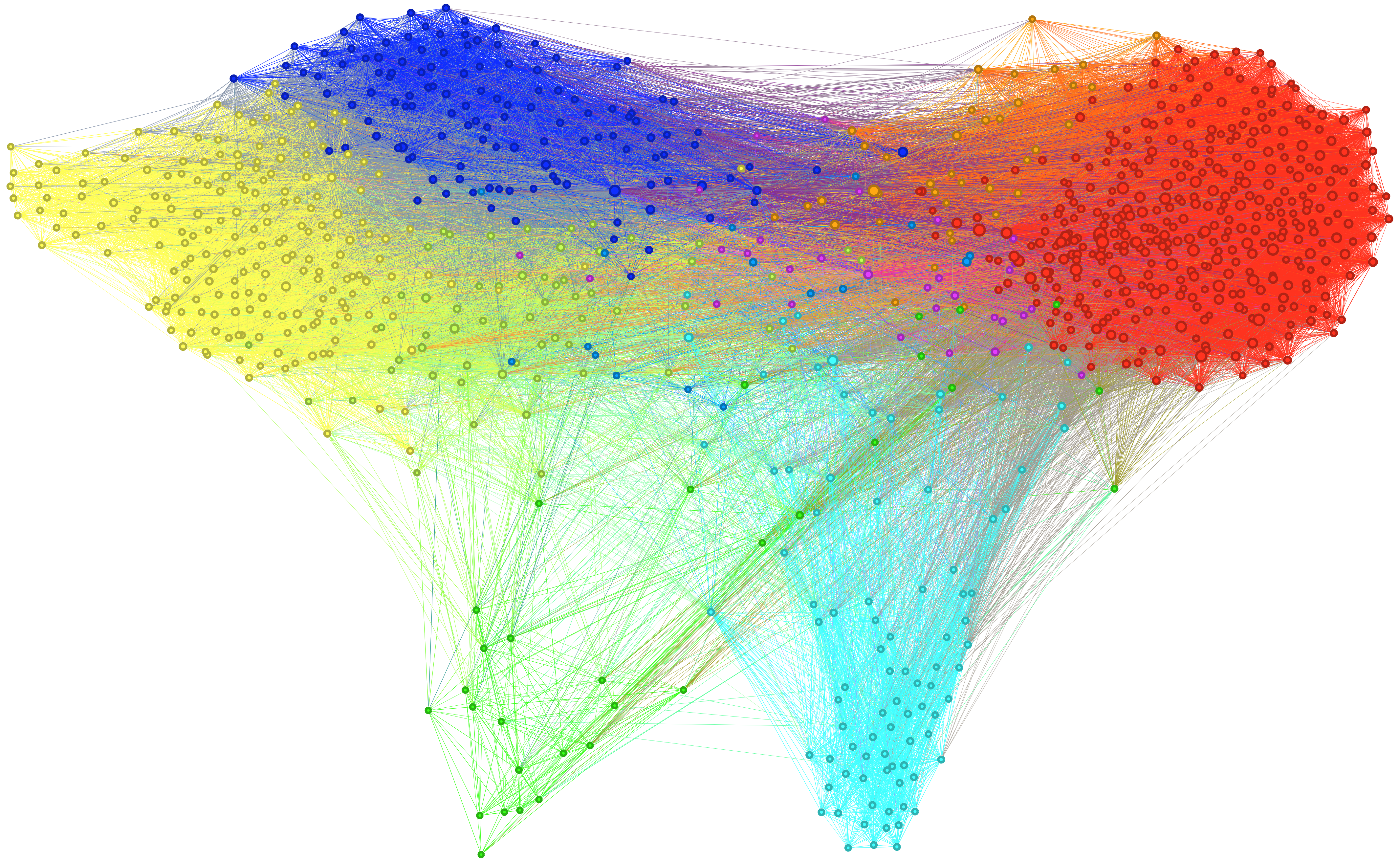
Graph

Settings of our problem

In most applications, the **exact** computation of triangles is unfeasible, due to the size of the data.

Settings of our problem

In most applications, the **exact** computation of triangles is unfeasible, due to the size of the data.

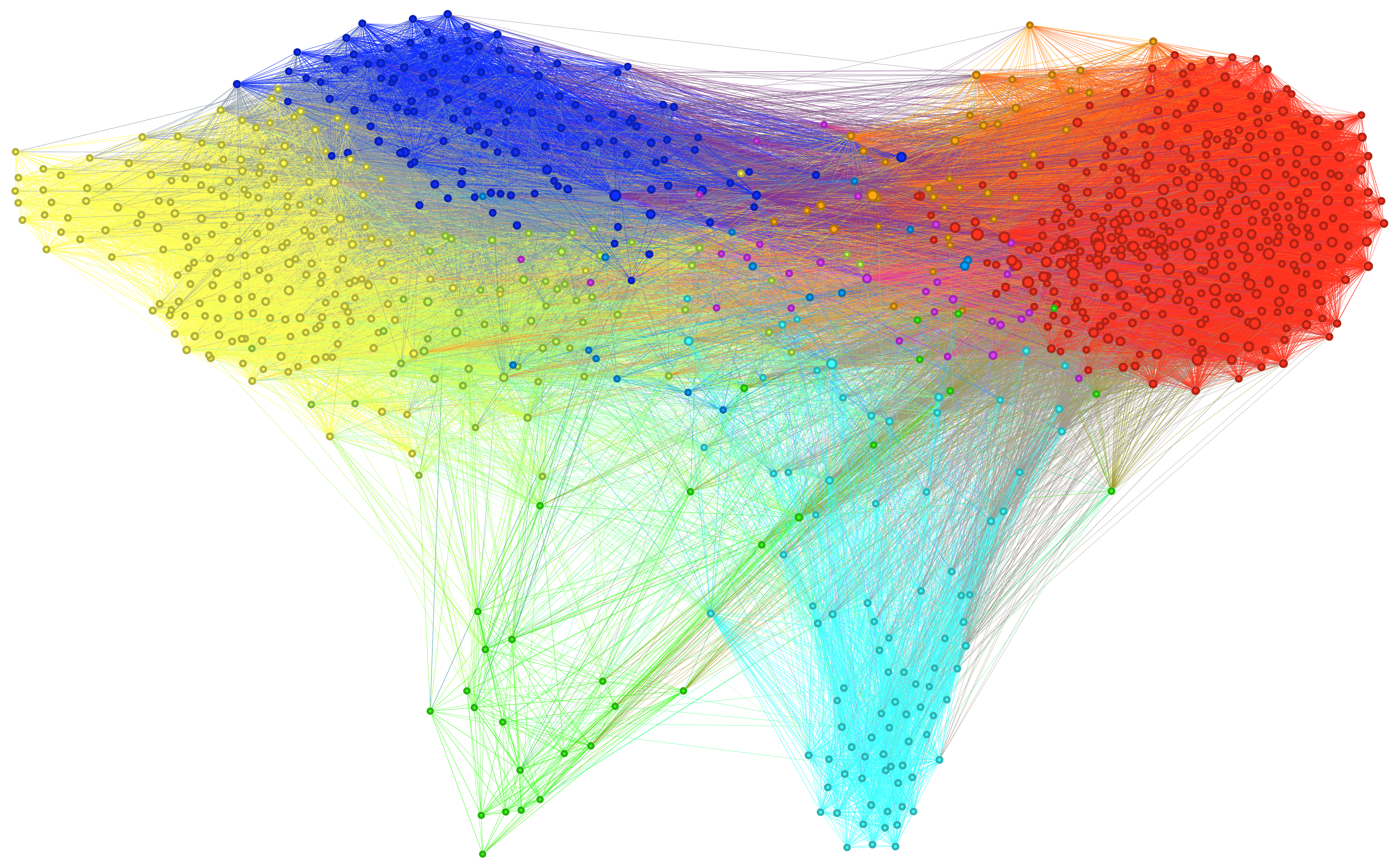


Graph of **Twitter** followers

Settings of our problem

In most applications, the **exact** computation of triangles is unfeasible, due to the size of the data.

- Design **fast** and **efficient** algorithms, that provide **high-quality** approximation

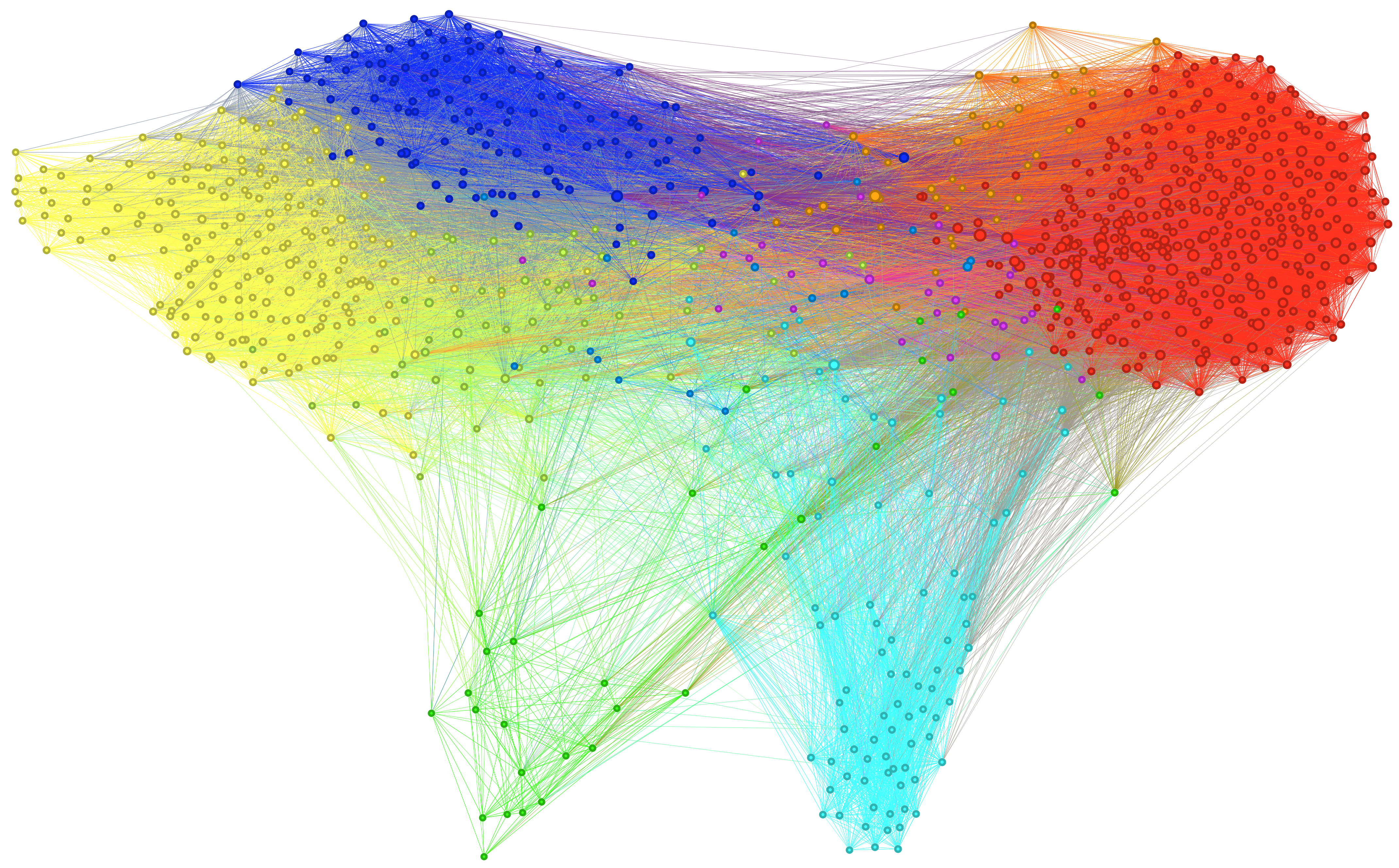


Graph of **Twitter** followers

Settings of our problem

In most applications, the **exact** computation of triangles is unfeasible, due to the size of the data.

- Design **fast** and **efficient** algorithms, that provide **high-quality** approximation
- For example, we can store a small fraction of edges of the graph



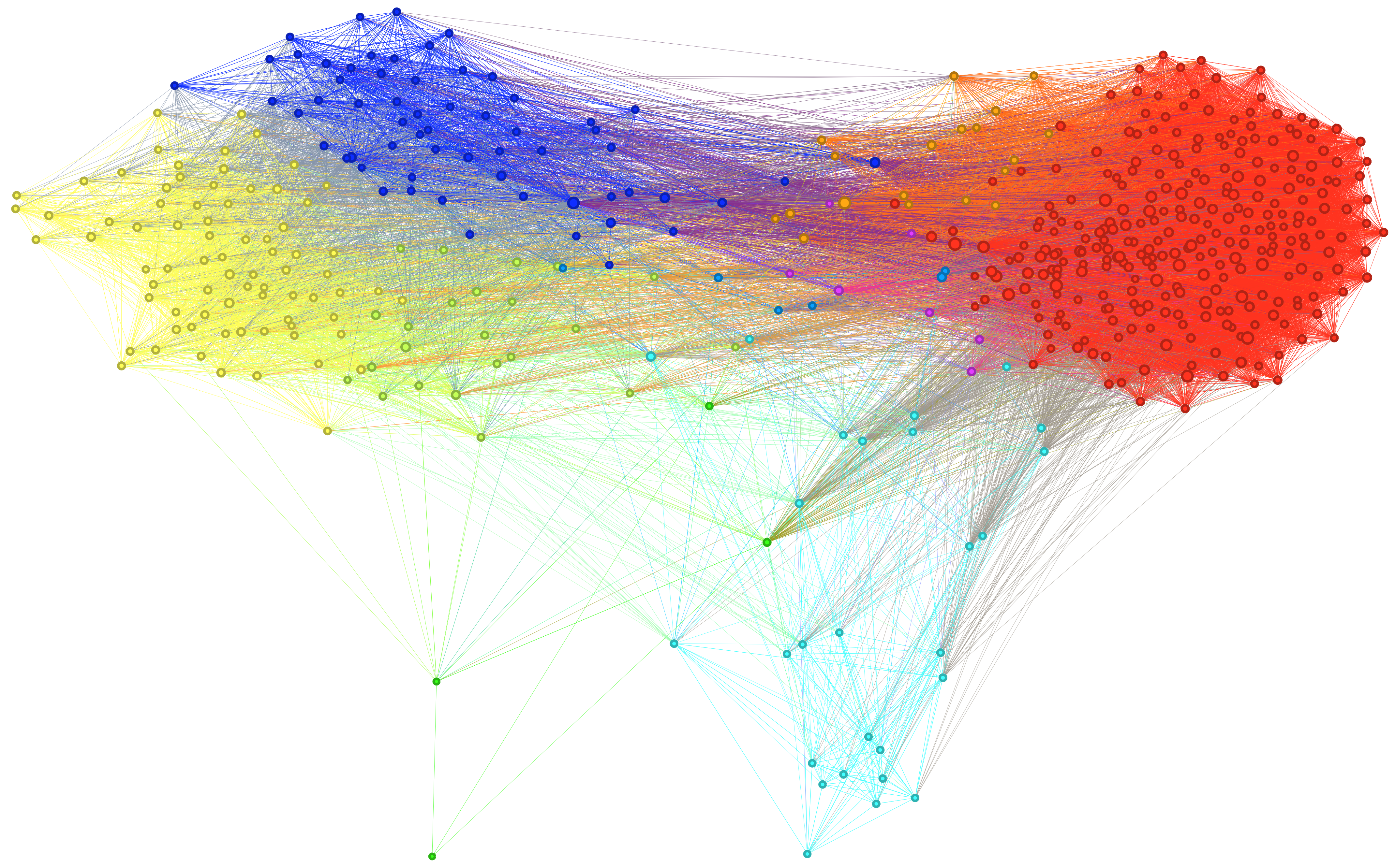
Graph of **Twitter** followers

Settings of our problem

In most applications, the **exact** computation of triangles is unfeasible, due to the size of the data.

- Design **fast** and **efficient** algorithms, that provide **high-quality** approximation
- For example, we can store a small fraction of edges of the graph

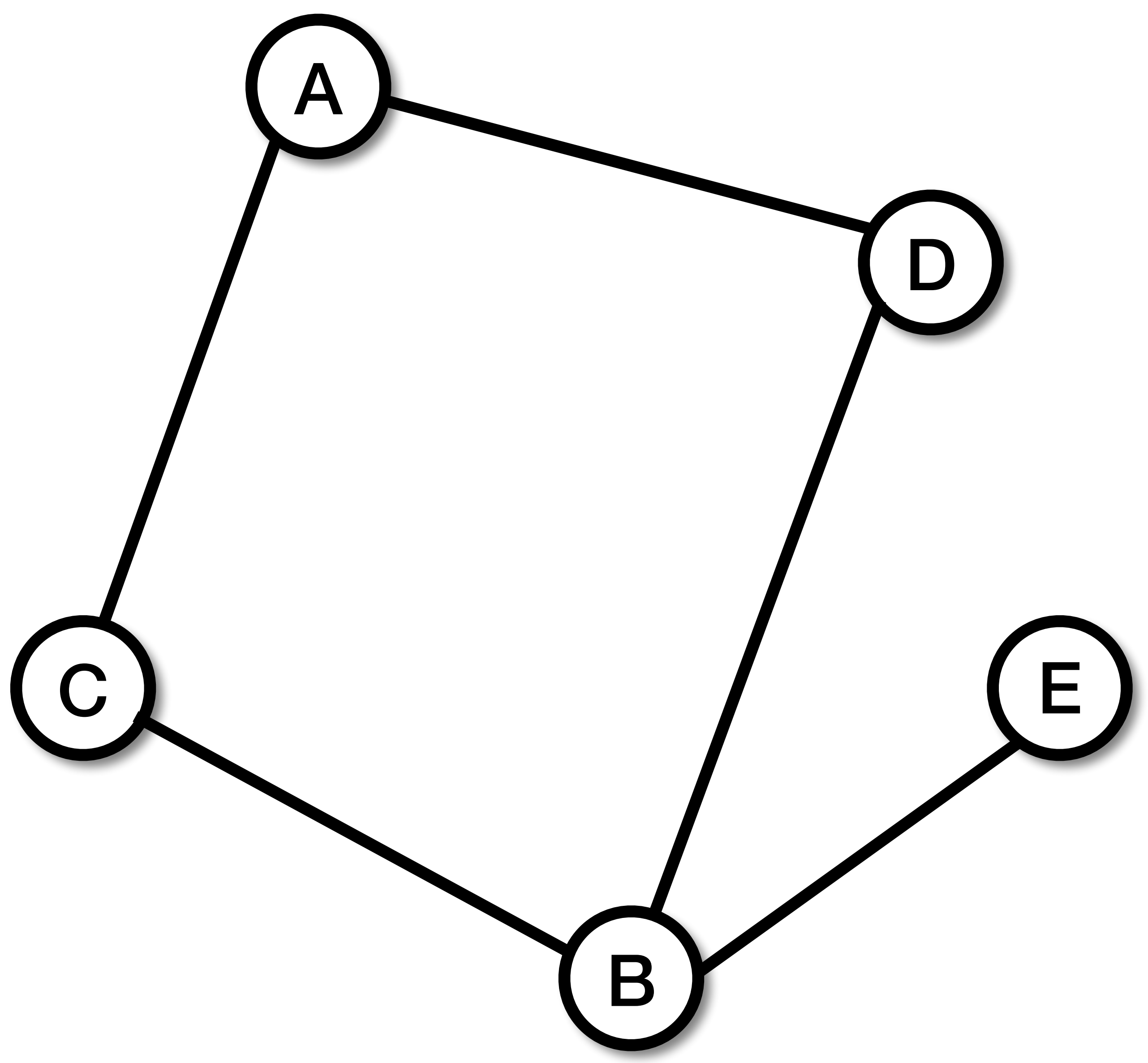
→ Use of **Sampling**



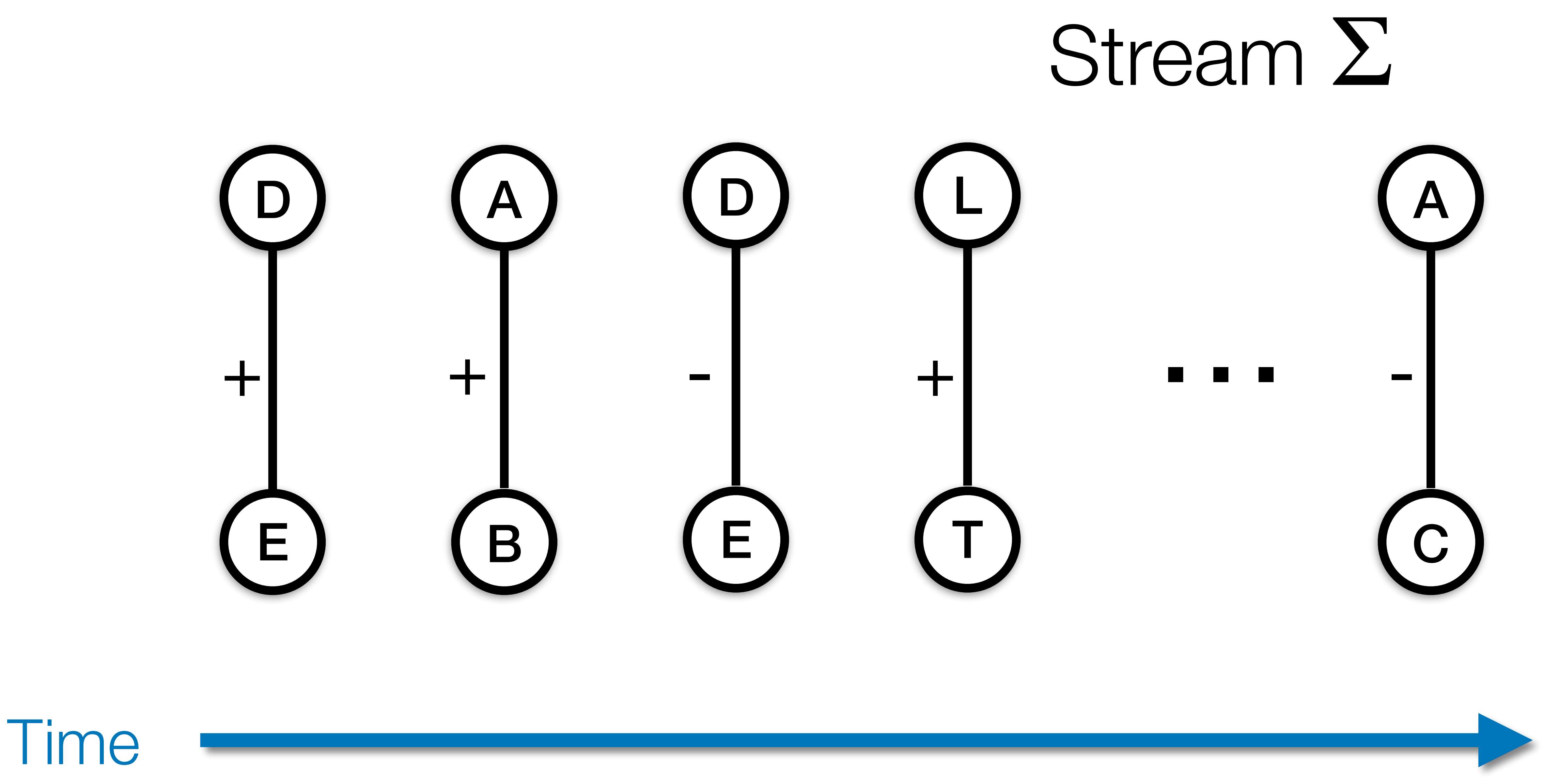
Sample of **Twitter** followers

Edge Sampling in Streaming

Each incoming edge on the stream is included in the sample with a certain probability.

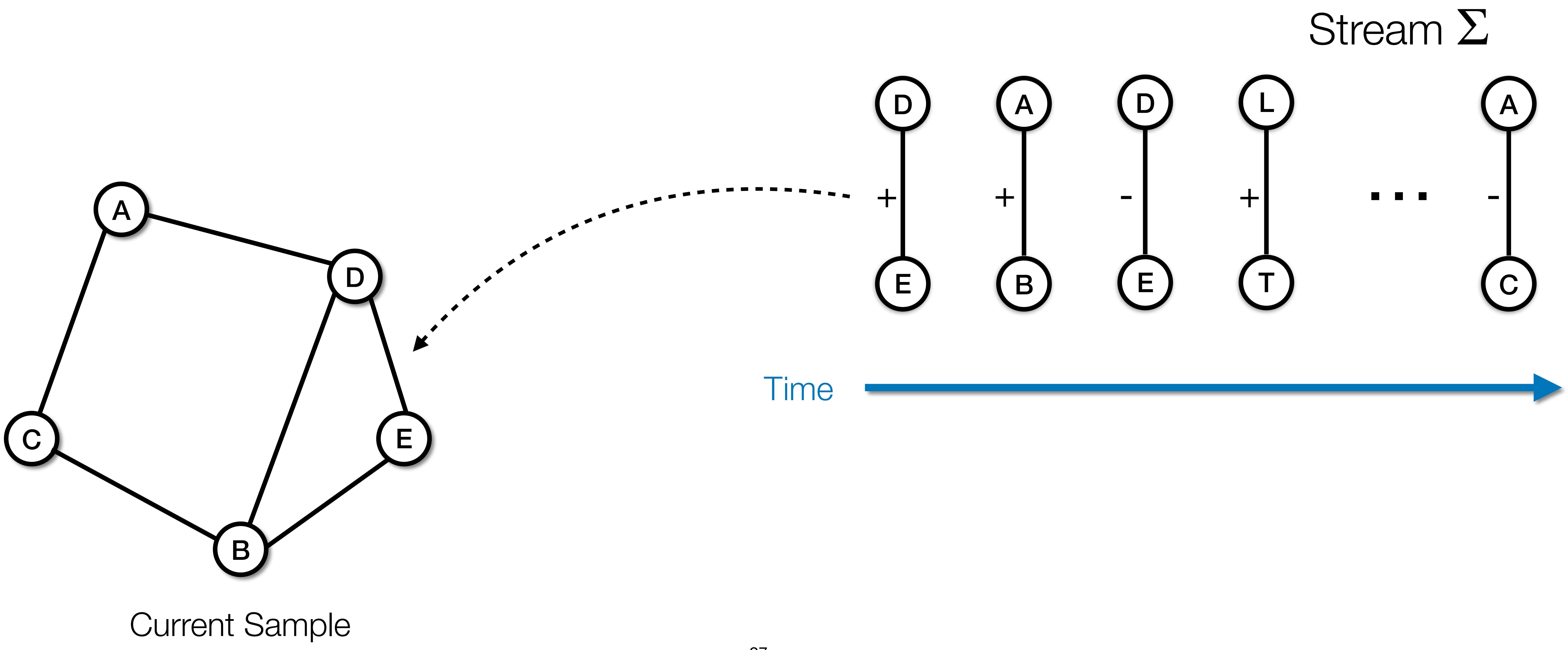


Current Sample



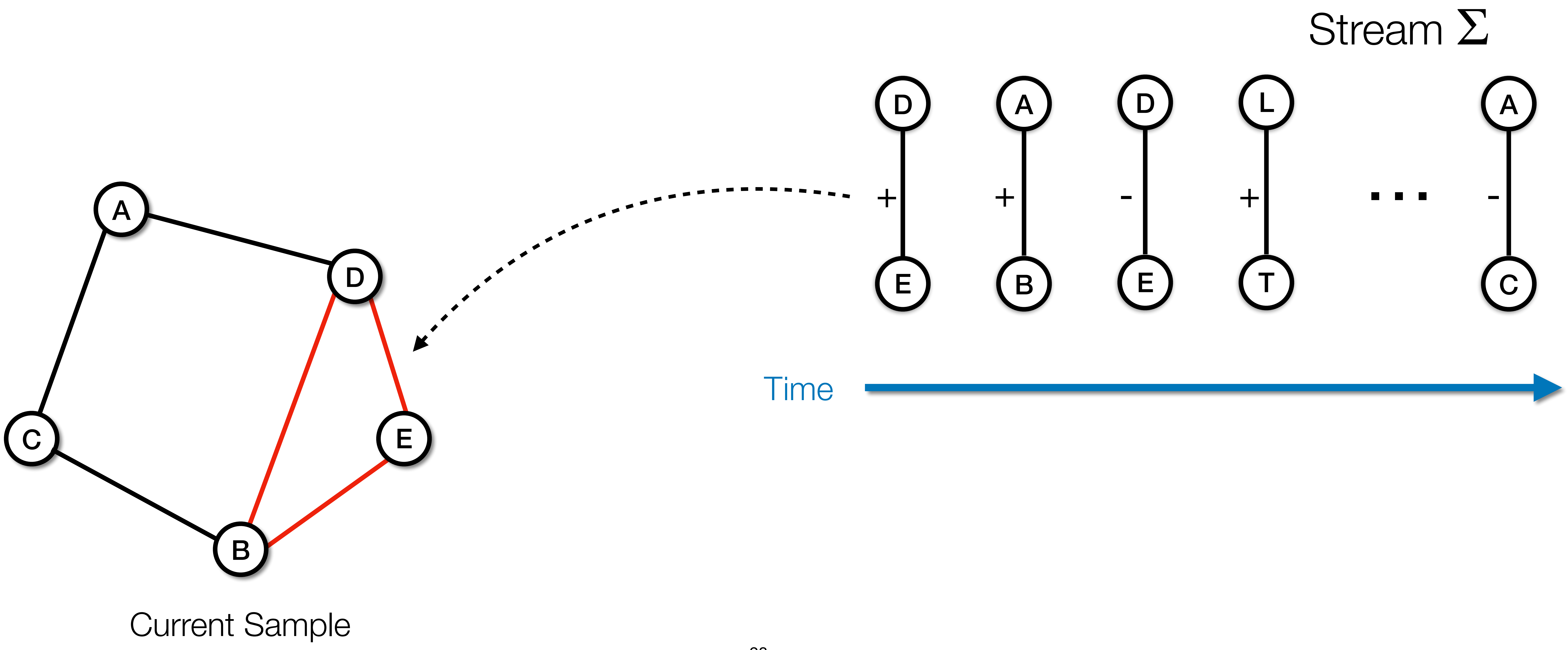
Edge Sampling in Streaming

Each incoming edge on the stream is included in the sample with a certain probability.



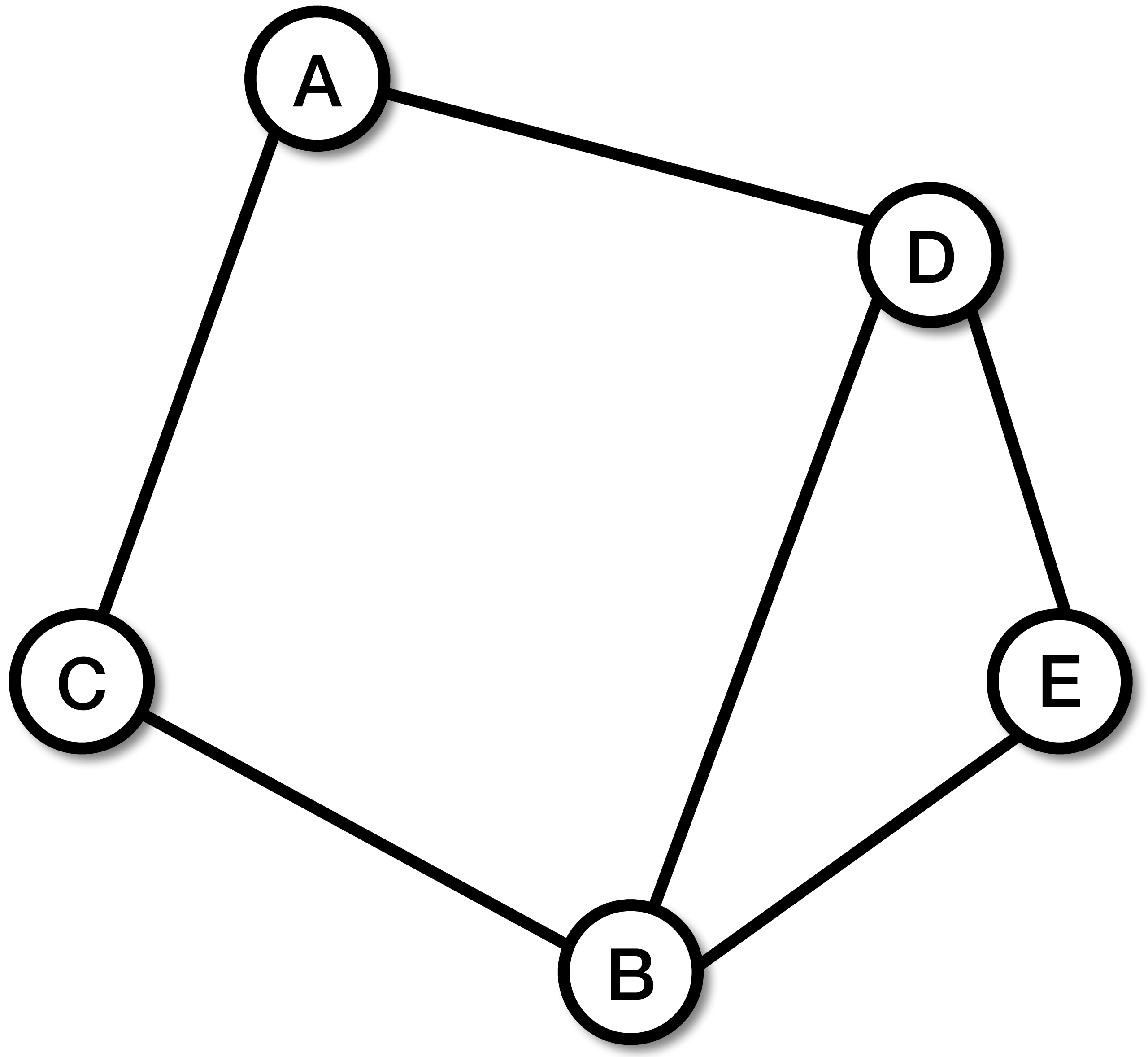
Edge Sampling in Streaming

Each incoming edge on the stream is included in the sample with a certain probability.



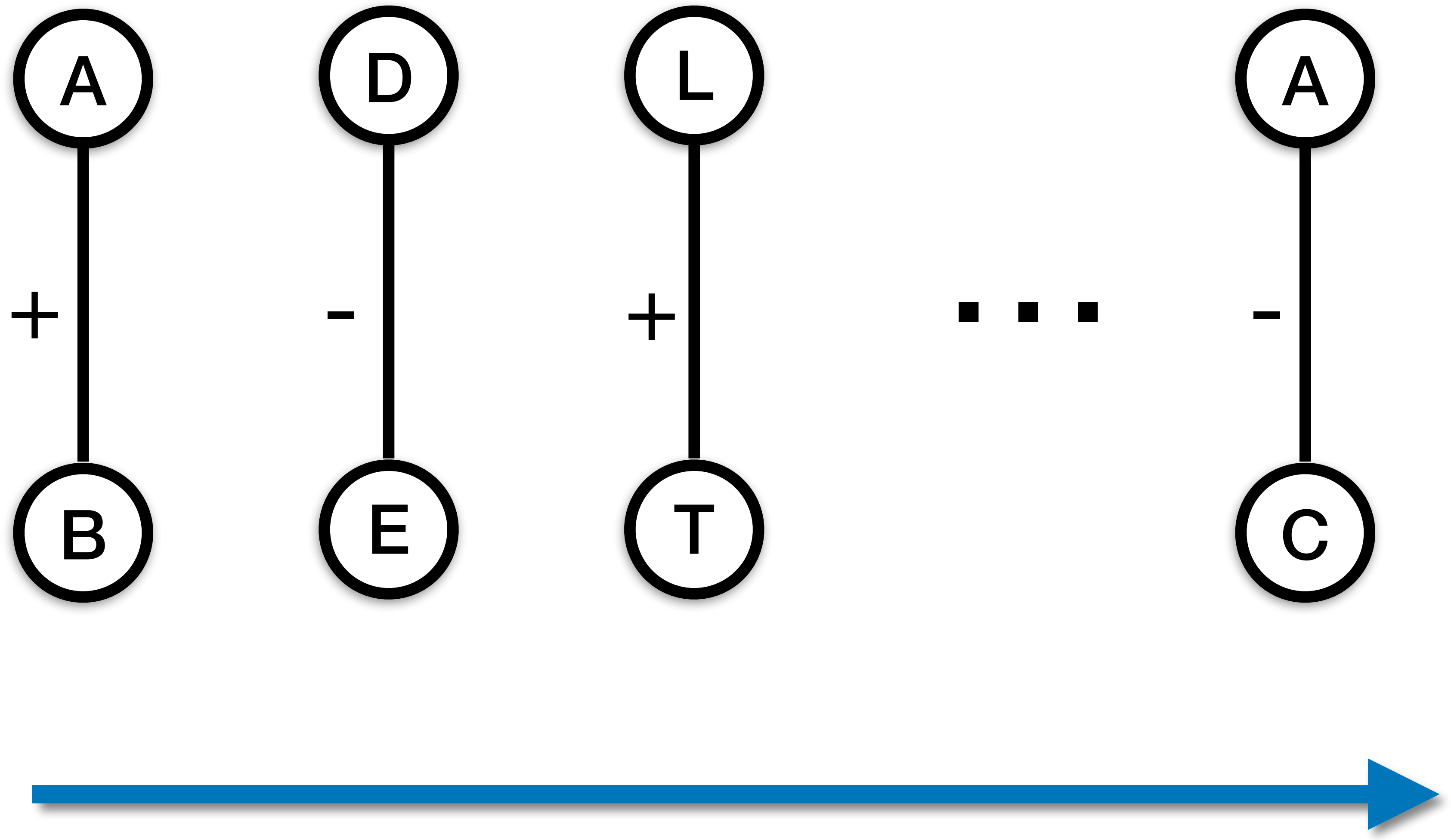
Edge Sampling in Streaming

Each incoming edge on the stream is included in the sample with a certain probability.



Current Sample

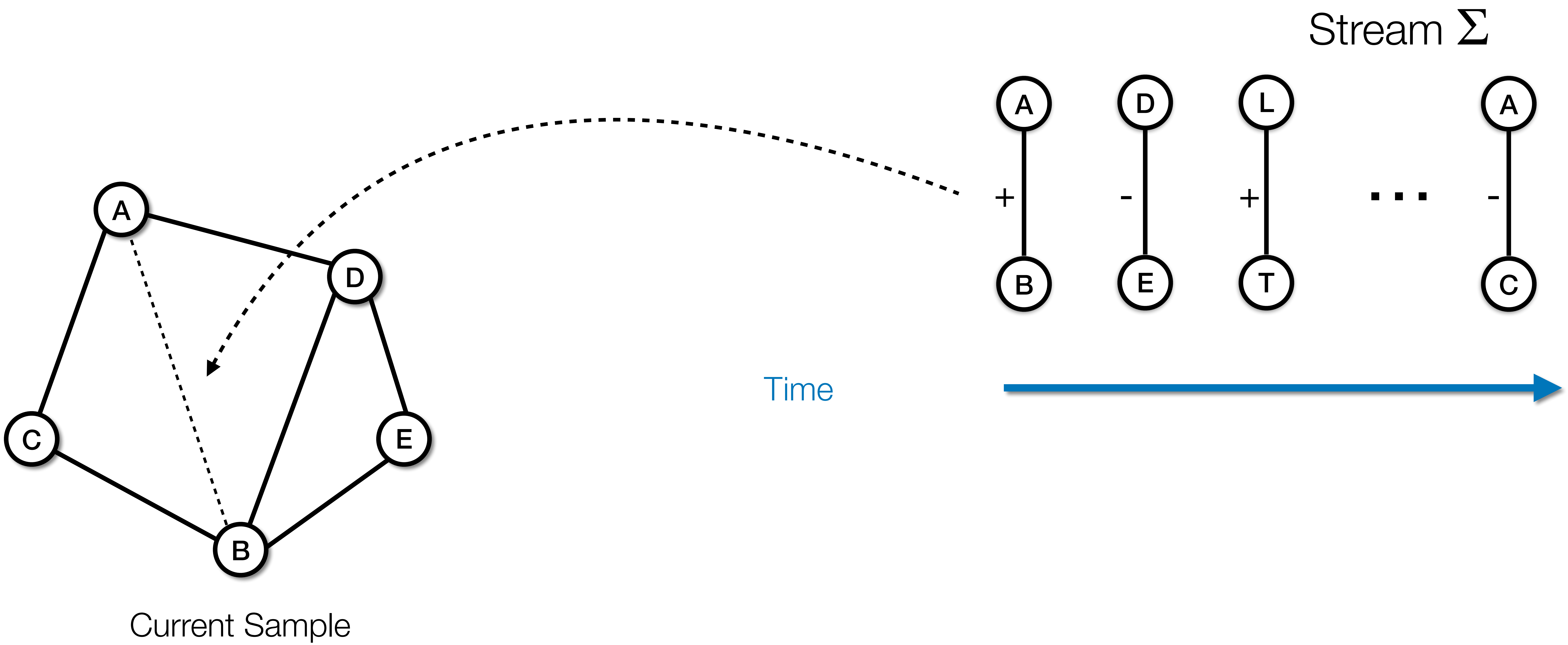
Stream Σ



Time

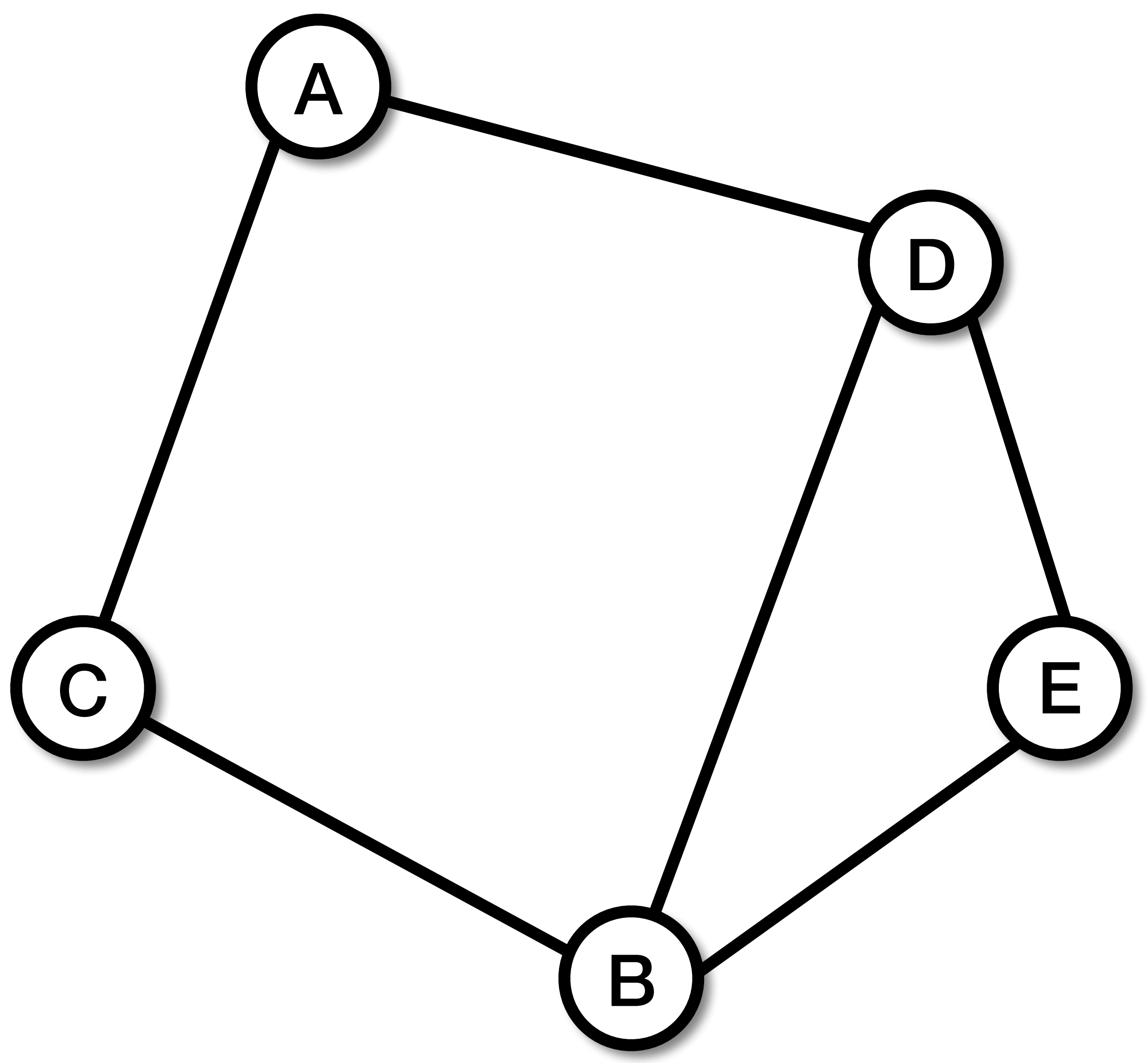
Edge Sampling in Streaming

Each incoming edge on the stream is included in the sample with a certain probability.

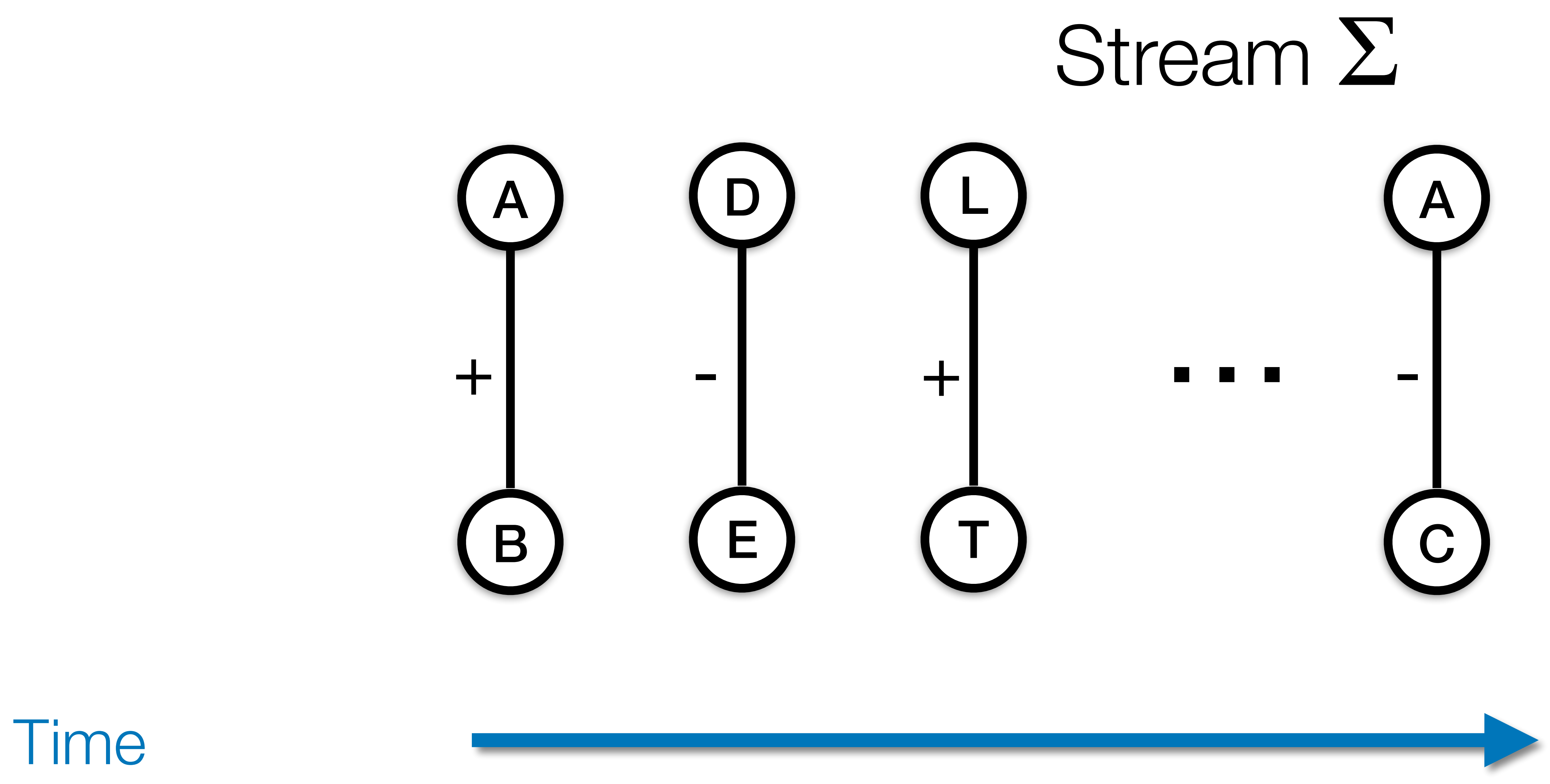


Edge Sampling in Streaming

Each incoming edge on the stream is included in the sample with a certain probability.



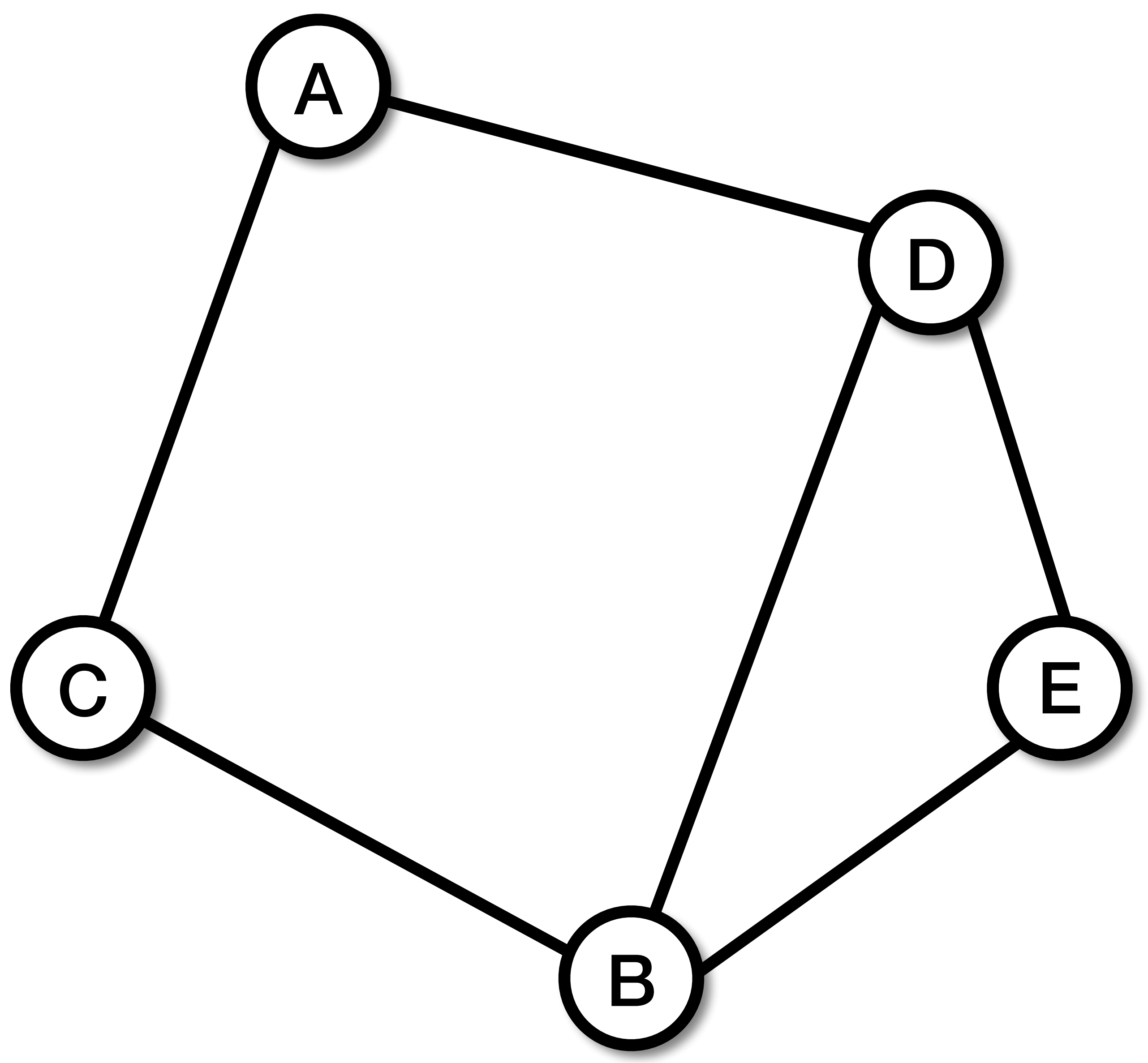
Current Sample



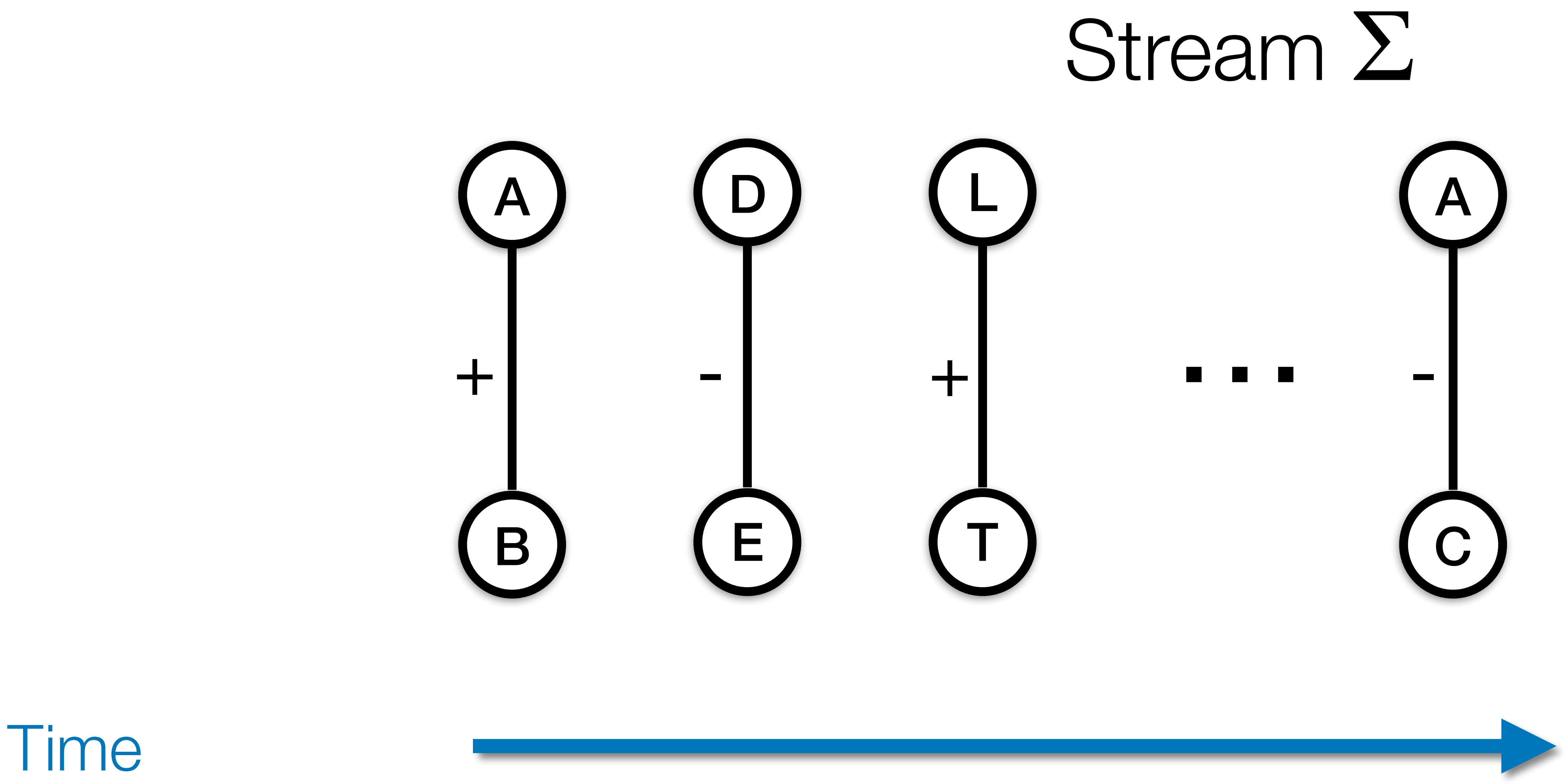
Edge Sampling in Streaming

Each incoming edge on the stream is included in the sample with a certain probability.

How to choose which edges to store?



Current Sample



State of The Art

For **insertion-only** streams, we consider*:*

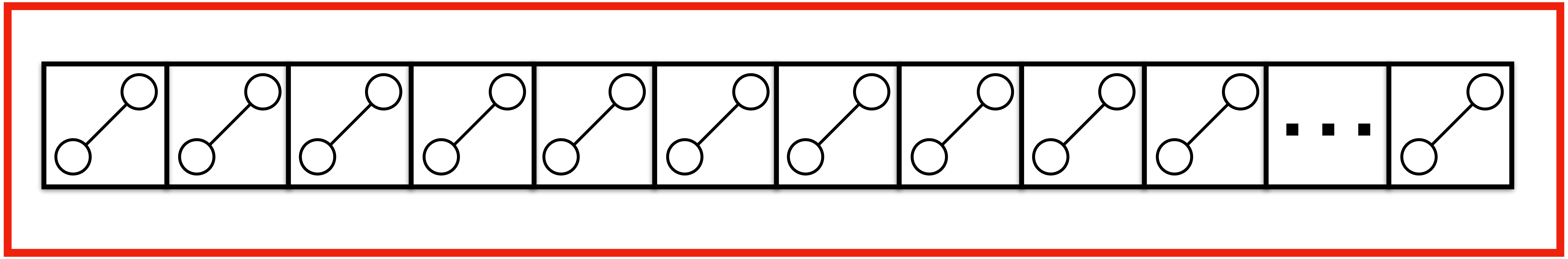
- ***Triest:*** [De Stefani et al., KDD 2016]
  Sample of edges via **reservoir sampling**

State of The Art

For **insertion-only** streams, we consider*:*

- ***Triest:*** [De Stefani et al., KDD 2016]
  Sample of edges via **reservoir sampling**

Uniform random sample of k edges



Memory budget k = number of edges to store

State of The Art

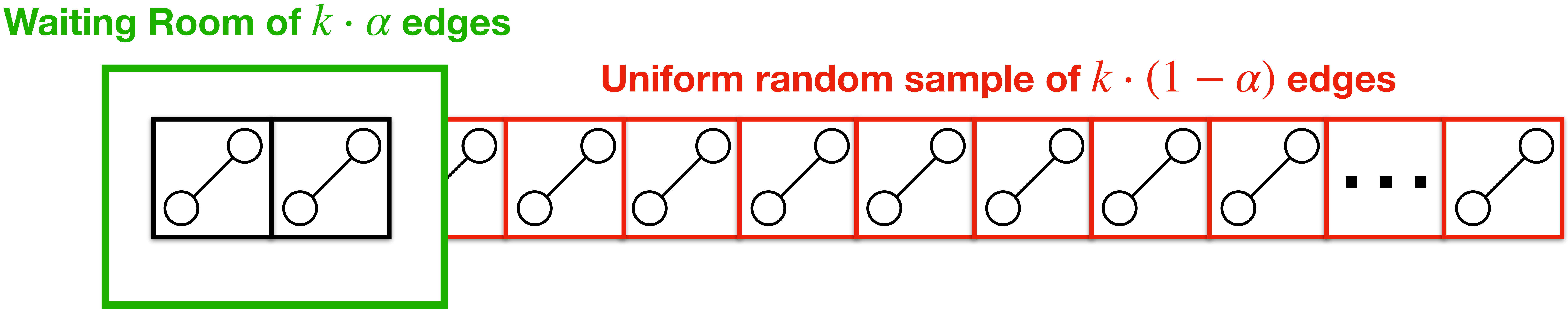
For **insertion-only** streams, we consider*:*

- ***WRS:*** [Shin K., ICDM 2017]
  Most recent edges (**waiting room**) + **reservoir sampling**
  Exploit **temporal localities** in real graph streams

State of The Art

For **insertion-only** streams, we consider:

- ***WRS:*** [Shin K., ICDM 2017]
  Most recent edges (**waiting room**) + **reservoir sampling**
  Exploit **temporal localities** in real graph streams



Memory budget k = number of edges to store

State of The Art

This talk: Triangle Counting Using Predictions

C. Boldrin and F. Vandin, “Fast and Accurate Triangle Counting in Graph Streams **Using Predictions**”, ICDM 2024

Algorithms with Predictions

Use of predictions about the input data has been formalised in the **“Algorithms with Predictions”** framework [Mitzenmacher and Vassilvitskii, 2020]

- Go beyond worst-case analysis
- Predictor empowering effectiveness of classical algorithms

State of The Art

For **insertion-only** streams, we consider*:*

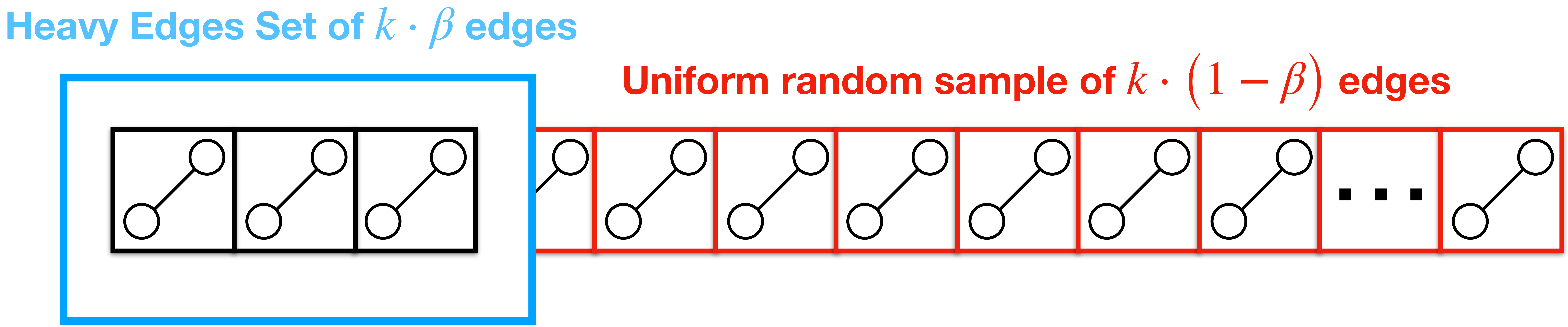
- ***Chen:*** [Chen et al., ICLR 2022]
  Heavy edges set + Fixed Probability Sampling

State of The Art

For **insertion-only** streams, we consider*:*

- ***Chen:*** [Chen et al., ICLR 2022]

  Heavy edges set + Fixed Probability Sampling

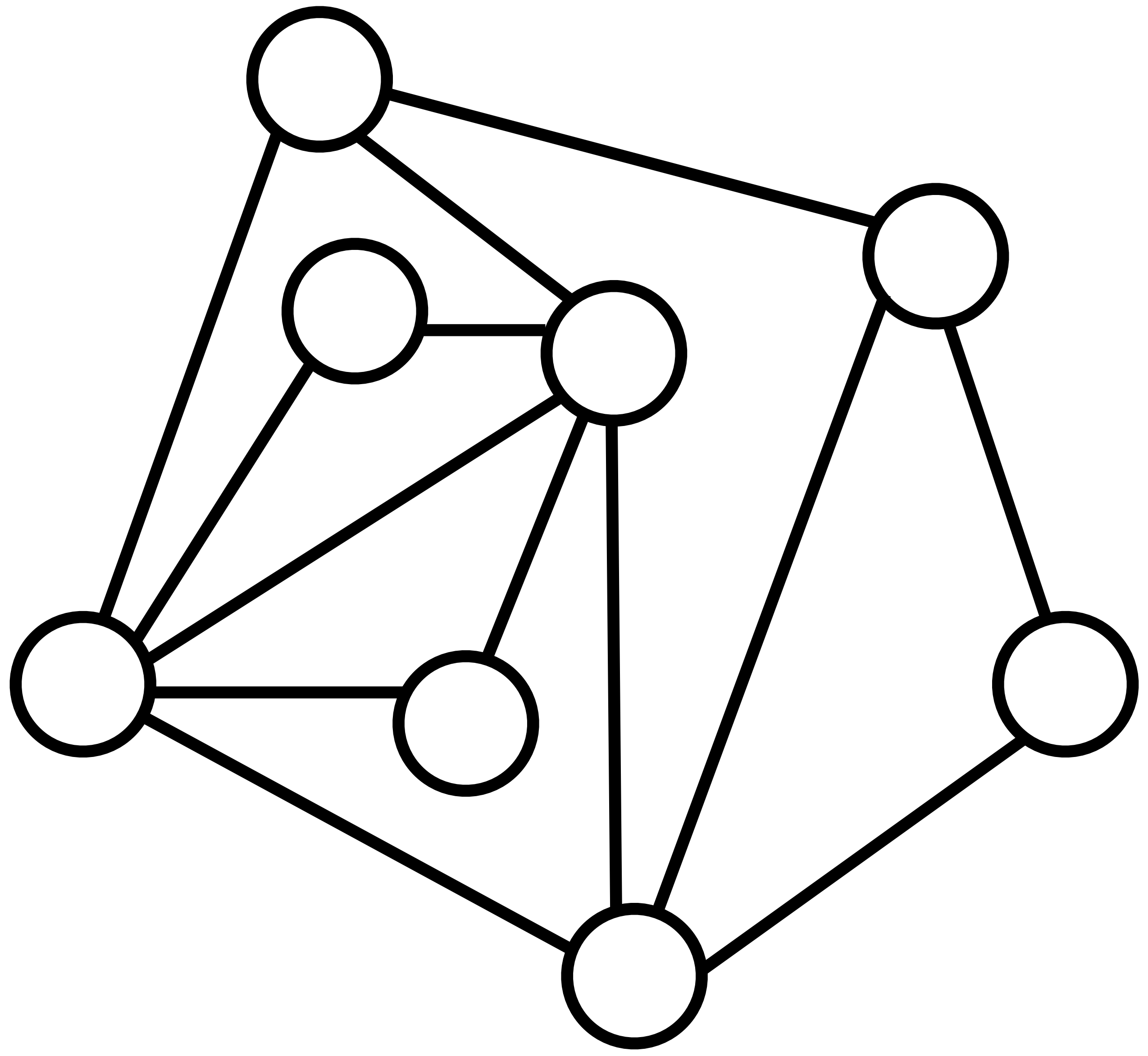


Memory budget k = number of edges to store

Heavy Edges

Heaviness of an edge e: number of triangles incident to e.

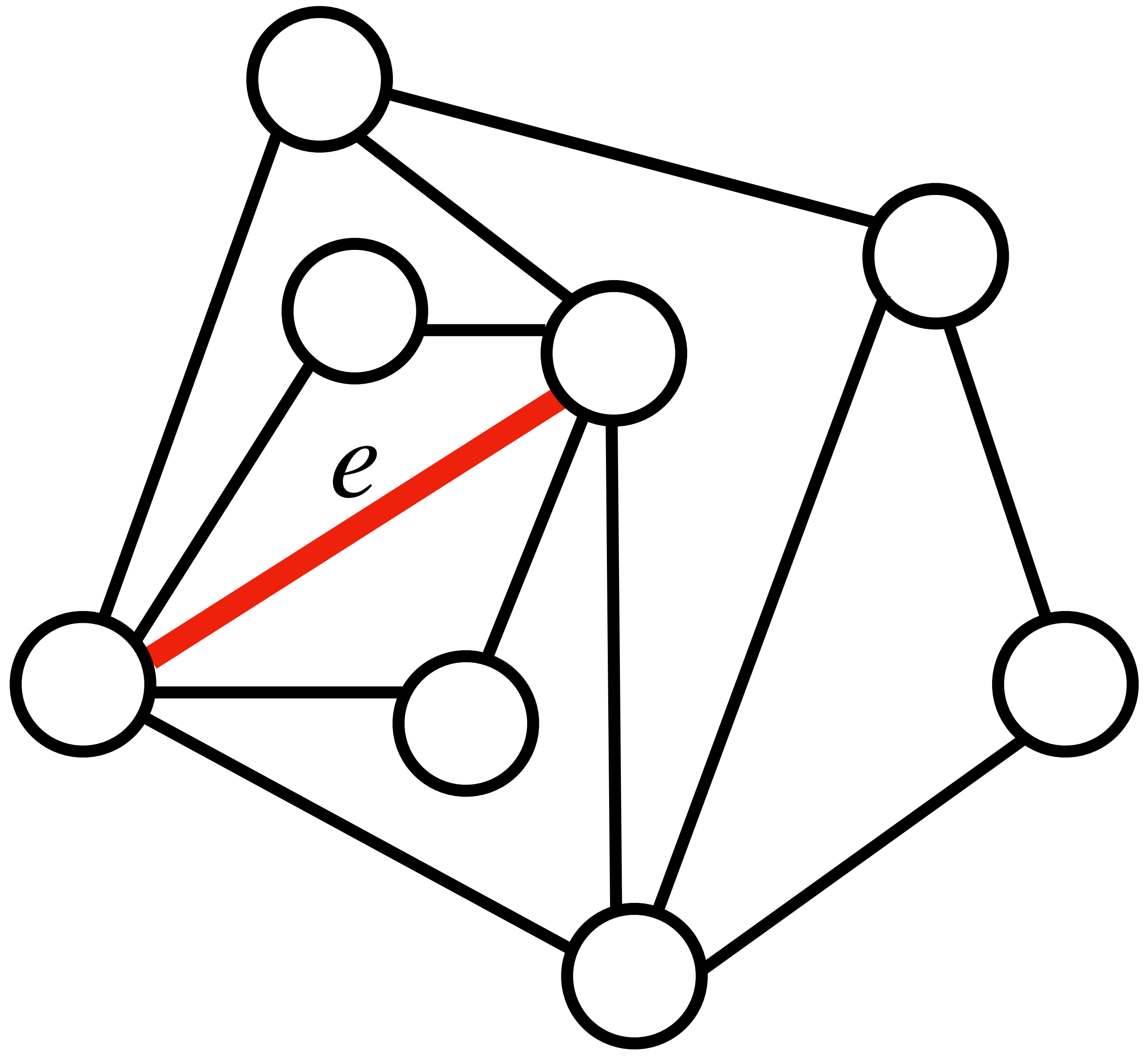
Idea: if an edge is heavy, we want to keep it in our sample.



Heavy Edges

Heaviness of an edge e: number of triangles incident to e.

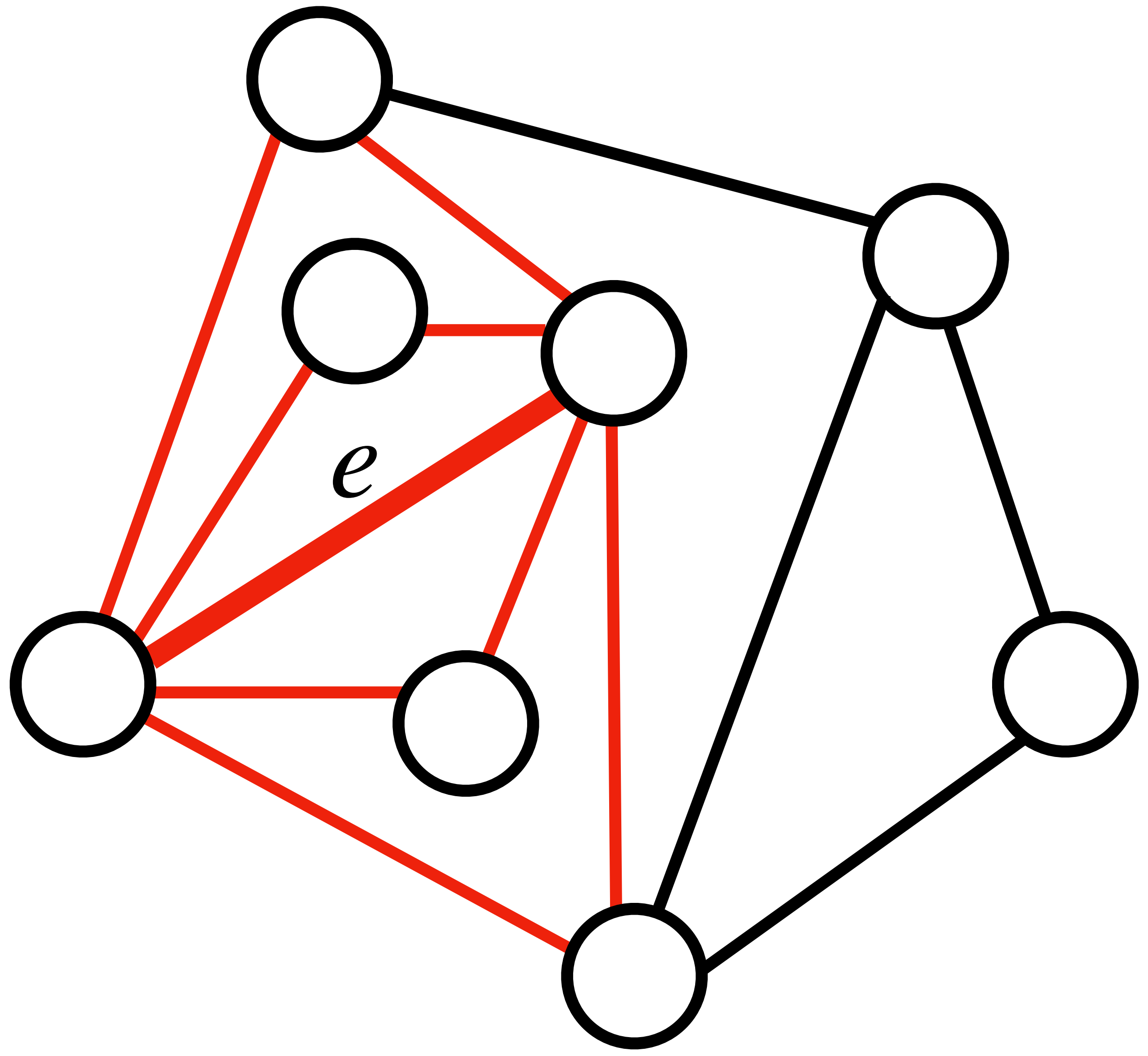
Idea: if an edge is heavy, we want to keep it in our sample.



Heavy Edges

Heaviness of an edge e: number of triangles incident to e.

Idea: if an edge is heavy, we want to keep it in our sample.

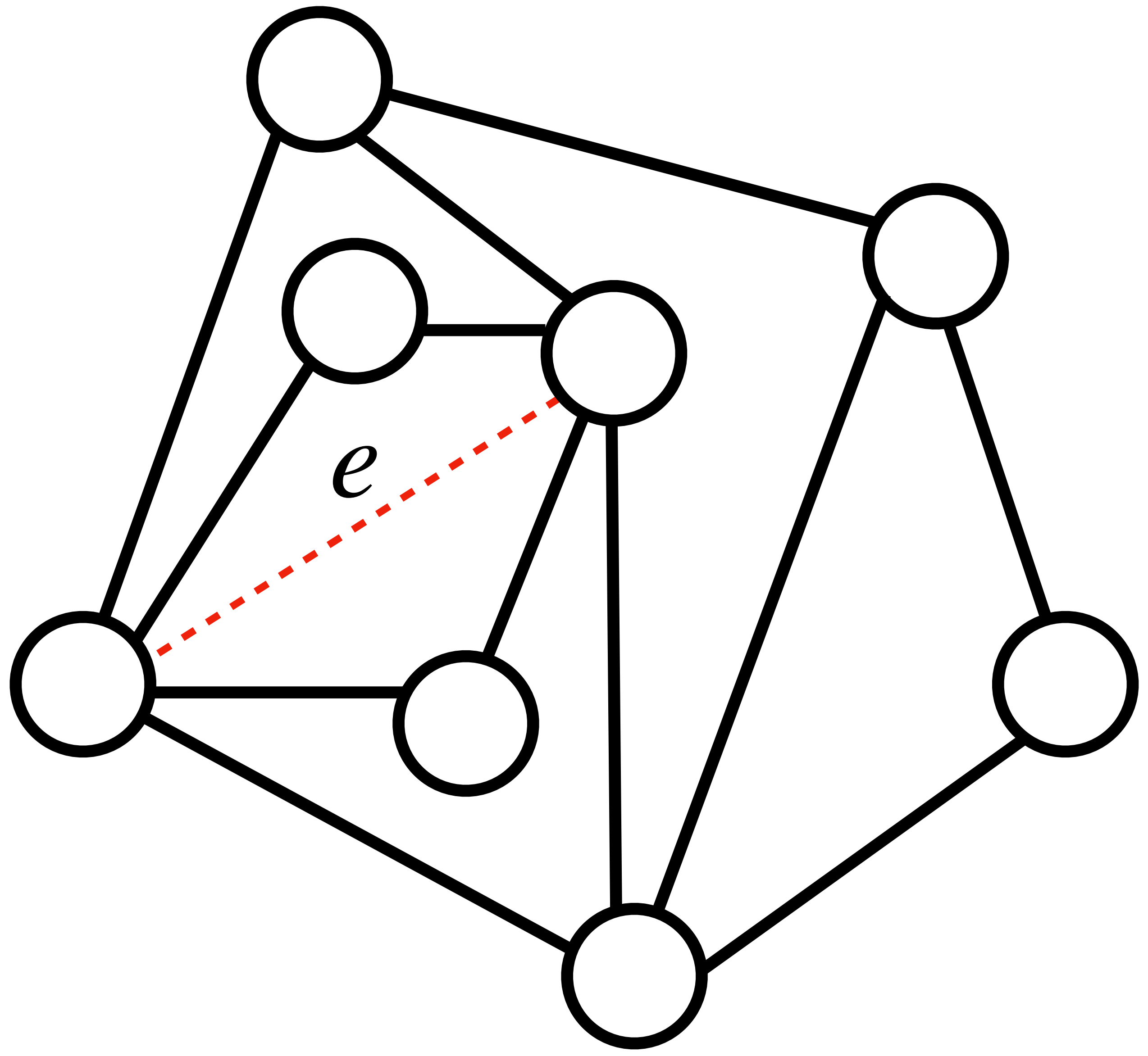


e is **heavy**, incident to
"many" triangles (4 triangles)

Heavy Edges

Heaviness of an edge e: number of triangles incident to e.

Idea: if an edge is heavy, we want to keep it in our sample.

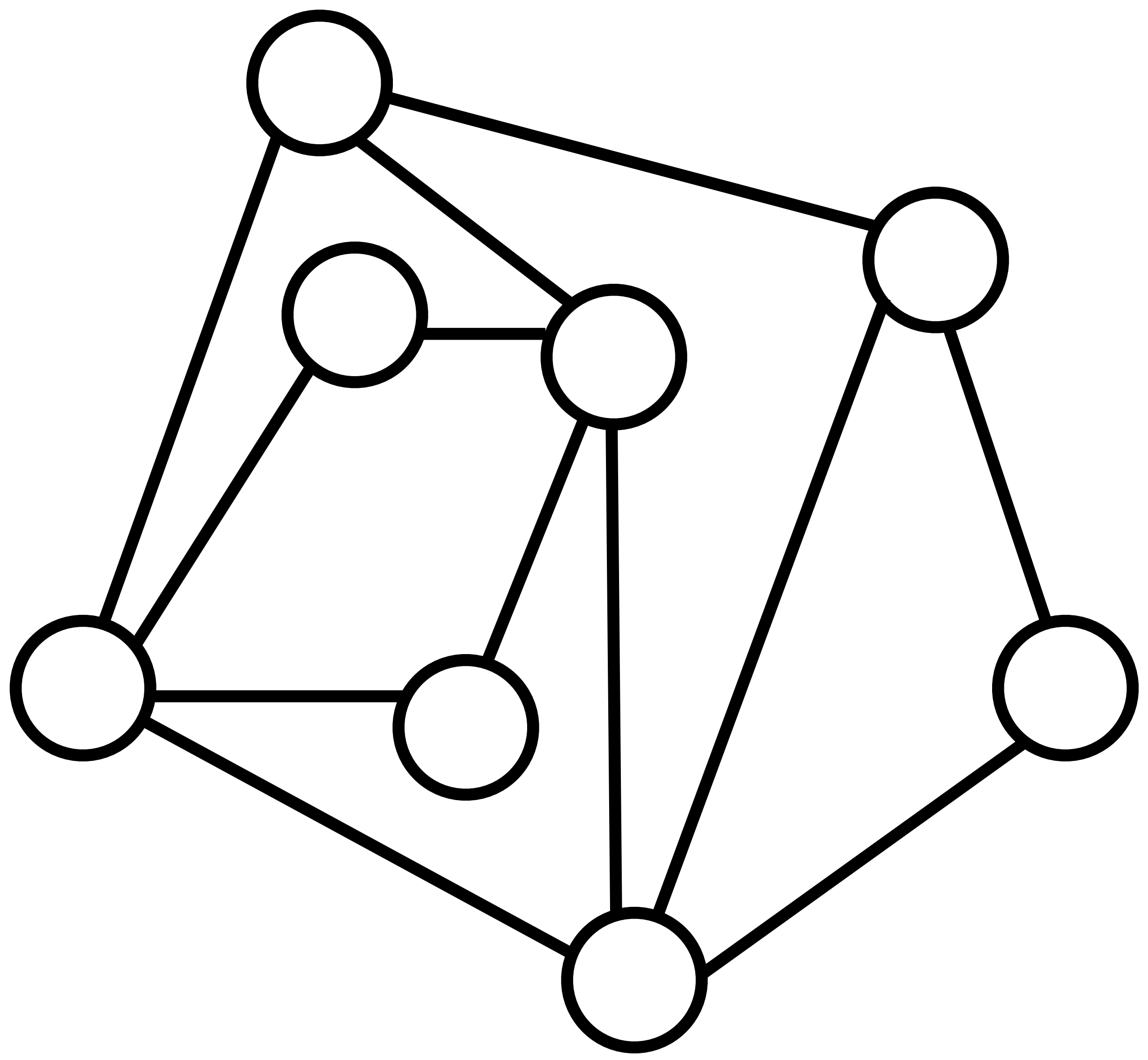


e is **heavy**, incident to
"many" triangles (4 triangles)

Heavy Edges

Heaviness of an edge e: number of triangles incident to e.

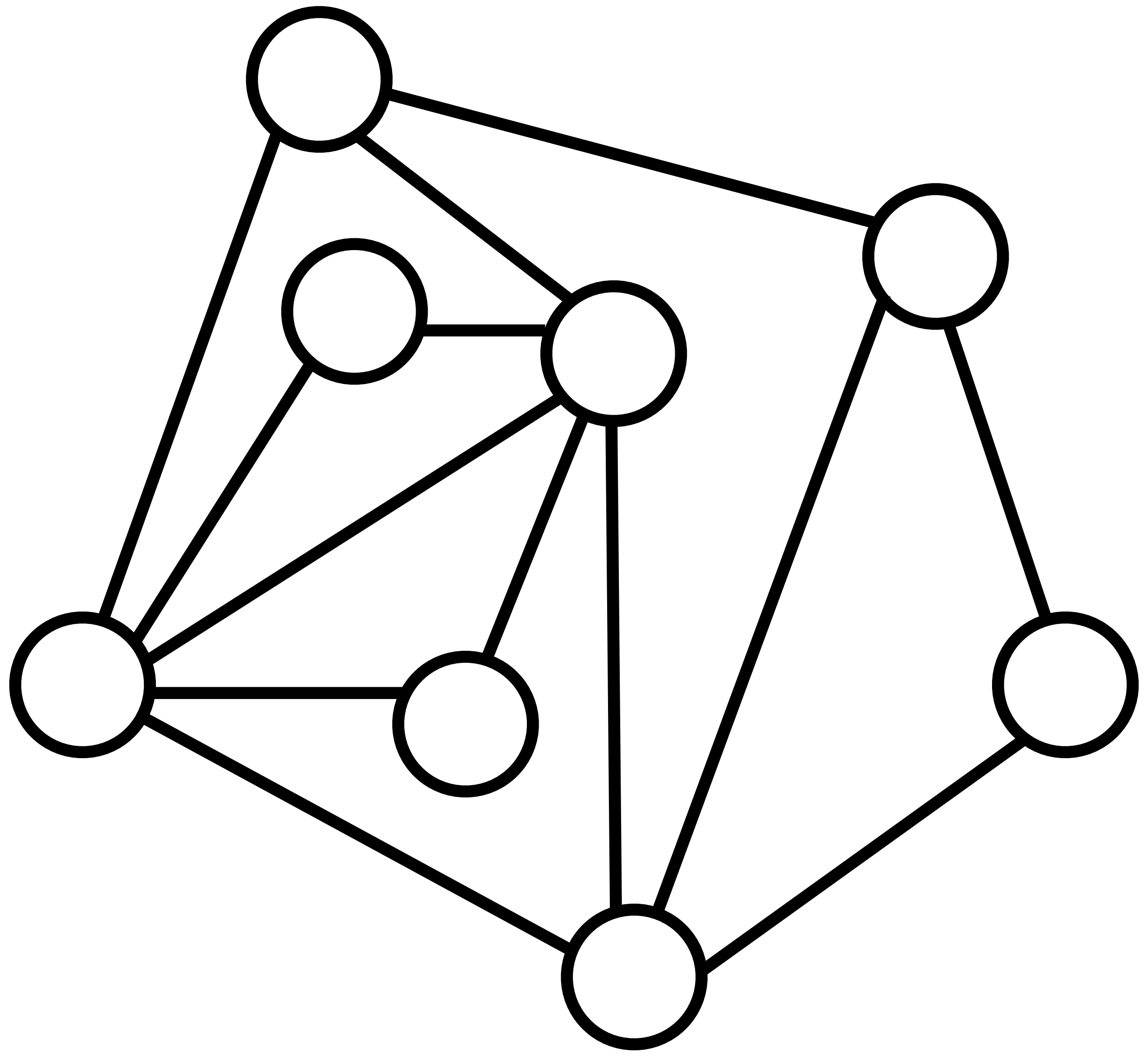
Idea: if an edge is heavy, we want to keep it in our sample.



Heavy Edges

Heaviness of an edge e: number of triangles incident to e.

Idea: if an edge is heavy, we want to keep it in our sample.



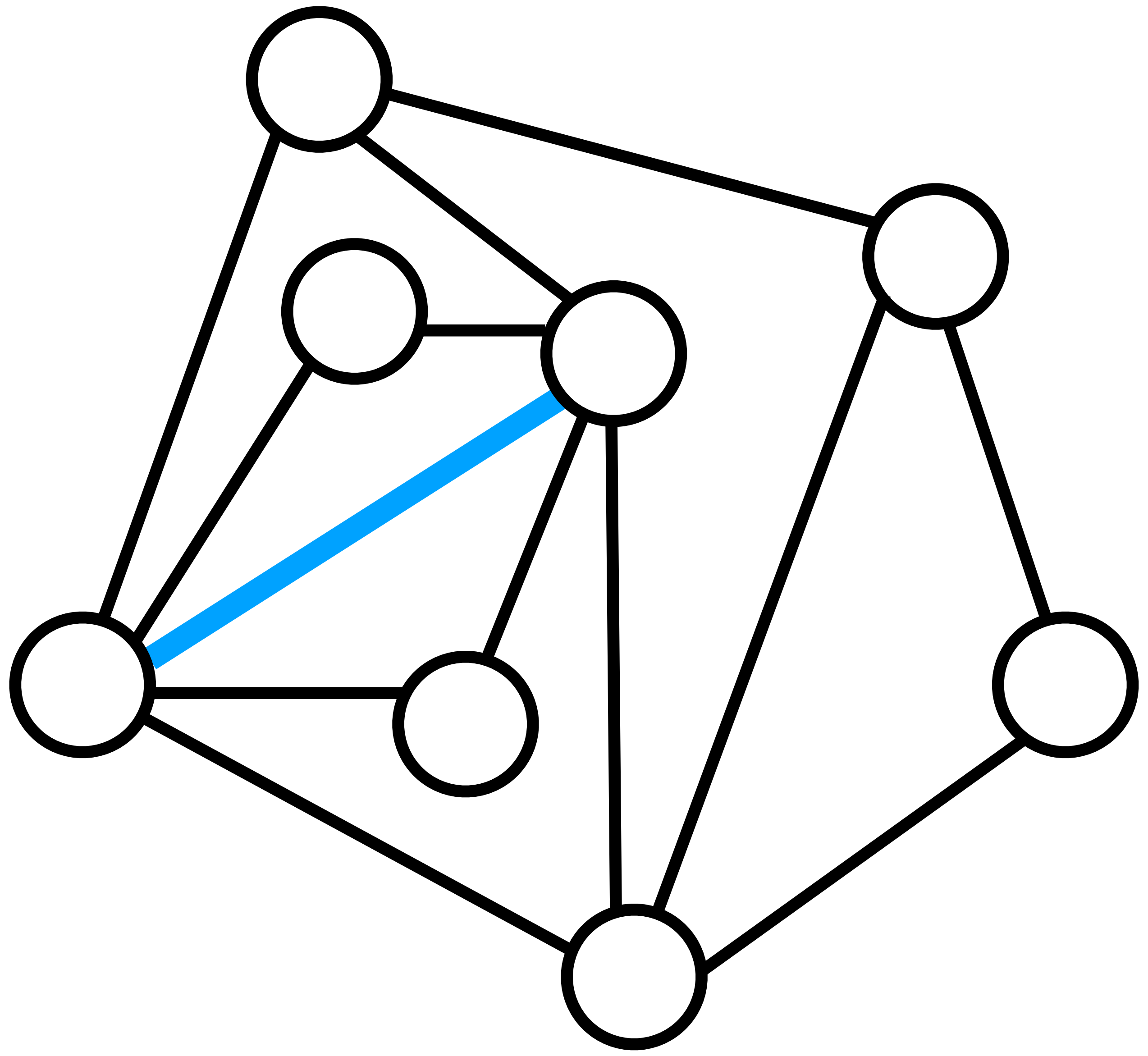
Assumption:

Predictor $O_H : E \rightarrow \mathbb{R}^+$ gives a measure *related* to the heaviness for each edge

Heavy Edges

Heaviness of an edge e: number of triangles incident to e.

Idea: if an edge is heavy, we want to keep it in our sample.



Assumption:

Predictor $O_H : E \to \mathbb{R}^+$ gives a measure *related* to the heaviness for each edge

Always store the **heaviest** edges in set H

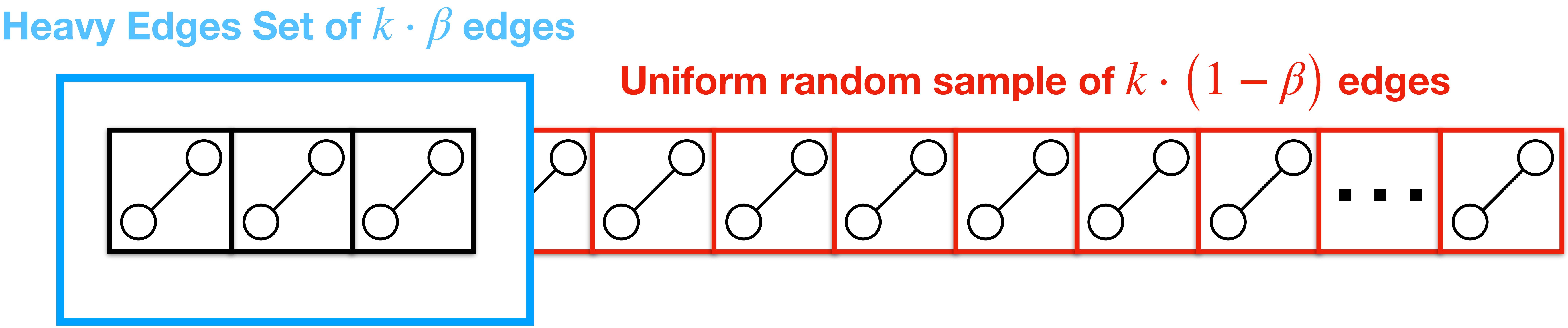
State of The Art

For **insertion-only** streams, we consider:

- ***Chen:*** [Chen et al., ICLR 2022]

  Heavy edges set + Fixed Probability Sampling

  Lack of **practical** predictor!



Memory budget k = number of edges to store

Challenges of Our Problem

Problem: Approximating the number of triangles in graph streams using predictions.

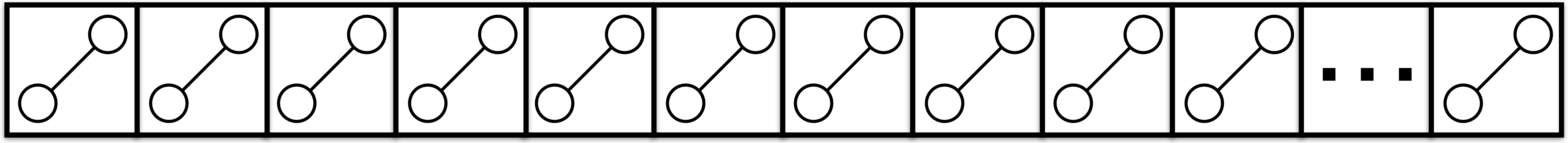
Challenges:

- Keep **high-quality** approximations **at every time** during the stream
- Do not exceed a given **memory budget**
- Updates of edges can only be **accessed once** (one-pass algorithm)
- Design a practical and efficient **predictor**

Overview of Our Algorithm

Our algorithm ***Tonic*** (Triangle cOuNting with predICtions) combines waiting room, heavy edges and uniform sampling

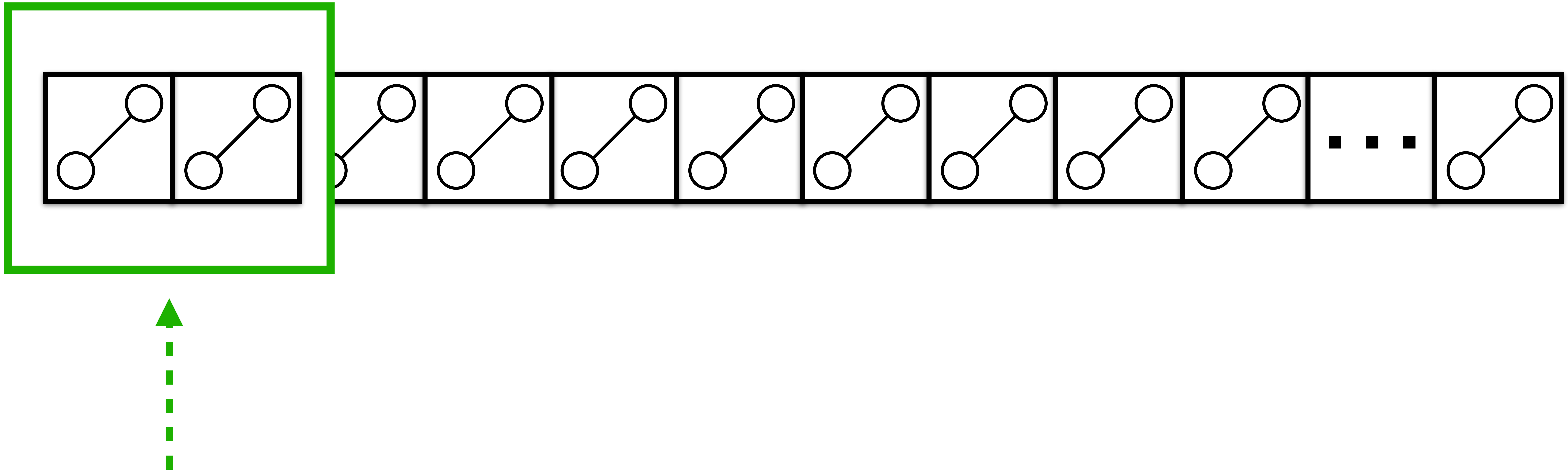
Memory budget k = number of edges to store



Overview of Our Algorithm

Our algorithm ***Tonic*** (Triangle cOuNting with predICtions) combines waiting room, heavy edges and uniform sampling

Memory budget k = number of edges to store

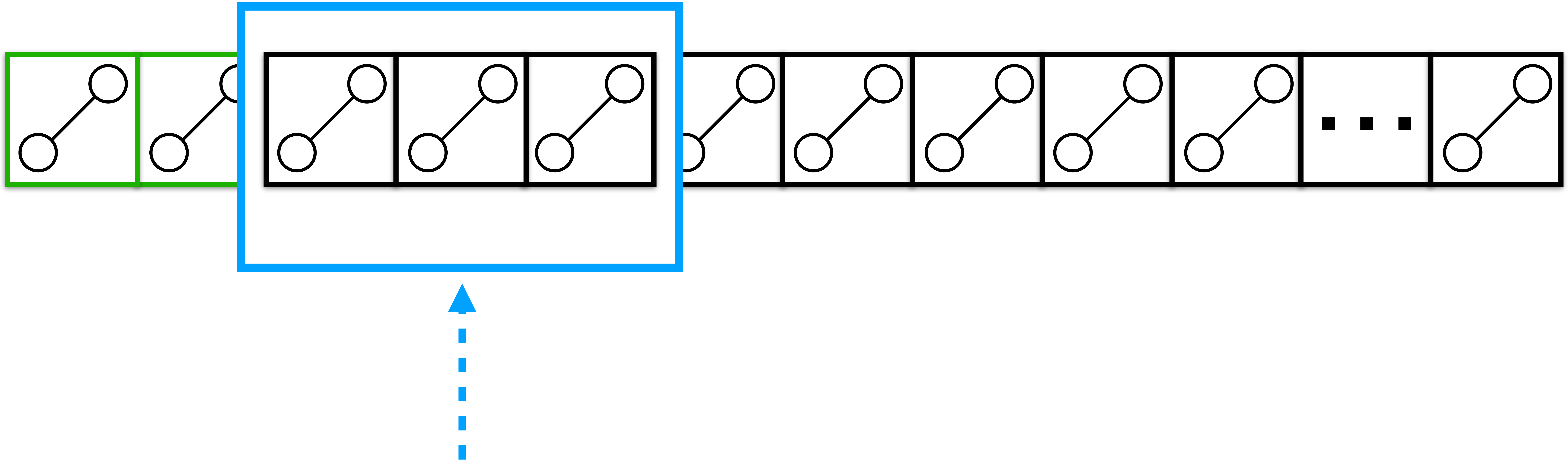


Store $k \cdot \alpha$ most recent edges in **waiting room** W

Overview of Our Algorithm

Our algorithm ***Tonic*** (Triangle cOuNting with predICtions) combines waiting room, heavy edges and uniform sampling

Memory budget k = number of edges to store



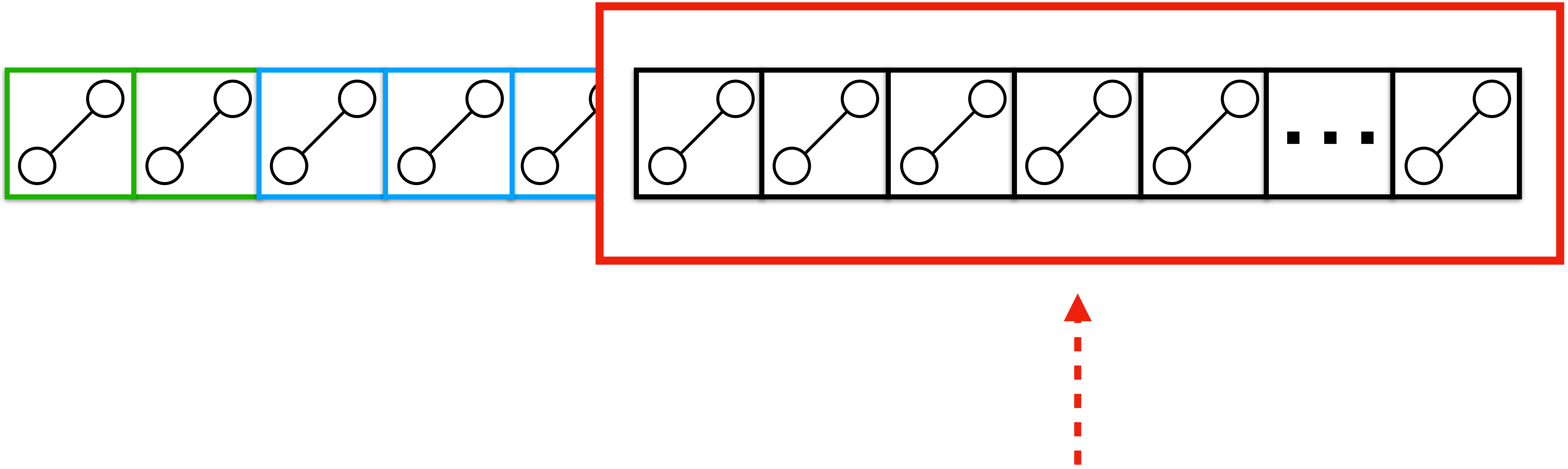
Store $k \cdot \alpha$ most recent edges in **waiting room W**

Store $k \cdot (1 - \alpha) \cdot \beta$ heaviest edges (according to the predictor) in **heavy edge set H**

Overview of Our Algorithm

Our algorithm ***Tonic*** (Triangle cOuNting with predICtions) combines waiting room, heavy edges and uniform sampling

Memory budget k = number of edges to store



Store $k \cdot \alpha$ most recent edges in **waiting room W**

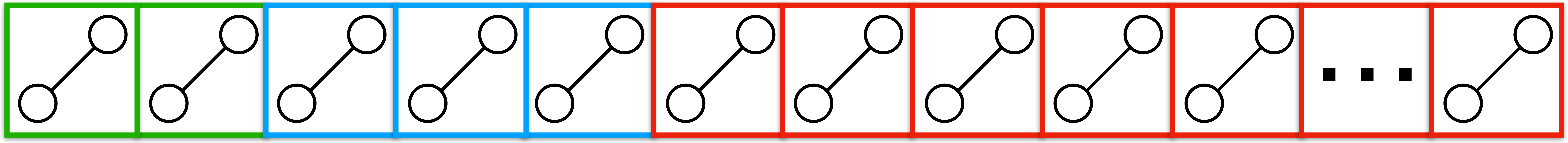
Store $k \cdot (1-\alpha) \cdot \beta$ heaviest edges (according to the predictor) in **heavy edge set H**

Store a **uniform random sample S** of $k \cdot (1-\alpha) \cdot (1-\beta)$ **light edges**

Overview of Our Algorithm

Our algorithm ***Tonic*** (Triangle cOuNting with predICtions) combines waiting room, heavy edges and uniform sampling

Memory budget k = number of edges to store



We empirically fix:
α = **0.05**, and β = **0.2**

Store $k \cdot \alpha$ most recent edges in **waiting room** W

Store $k \cdot (1 - \alpha) \cdot \beta$ heaviest edges (according to the predictor) in **heavy edge set** H

Store a **uniform random sample** S of $k \cdot (1 - \alpha) \cdot (1 - \beta)$ **light edges**

Our Contributions

- *Tonic* provides **fast** and **accurate** approximations of global (and local) triangles in both insertion-only and fully-dynamic graph streams
- We propose a **simple** and application-independent **predictor**, based on the **degree** of the nodes
- Extensive experimental evaluation shows improvements and scalability of *Tonic* with respect to the state of the art

Our Contributions

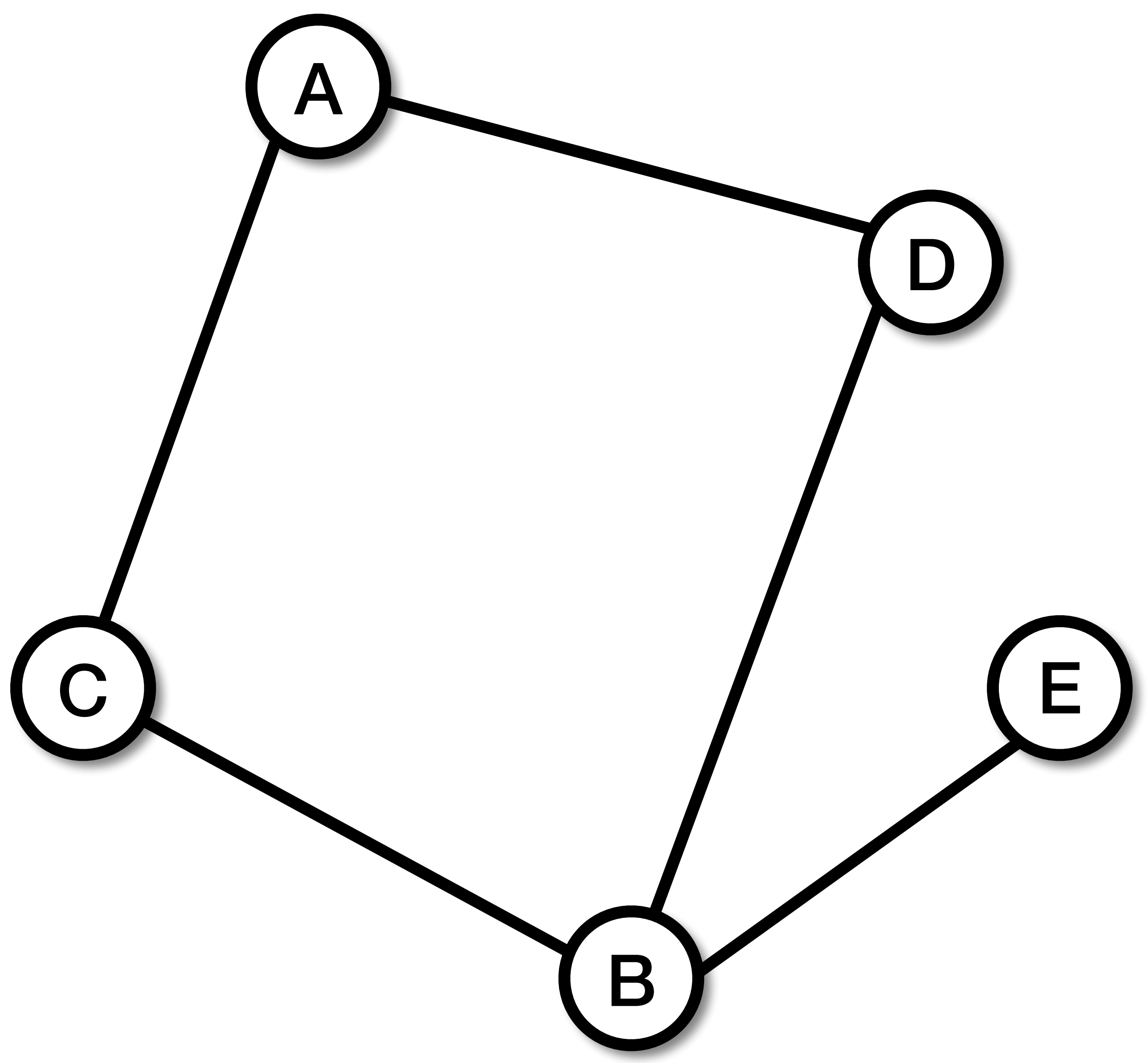
- *Tonic* provides **fast** and **accurate** approximations of **global** (and local) triangles in both **insertion-only** and fully-dynamic graph streams

- We propose a **simple** and application-independent **predictor**, based on the **degree** of the nodes

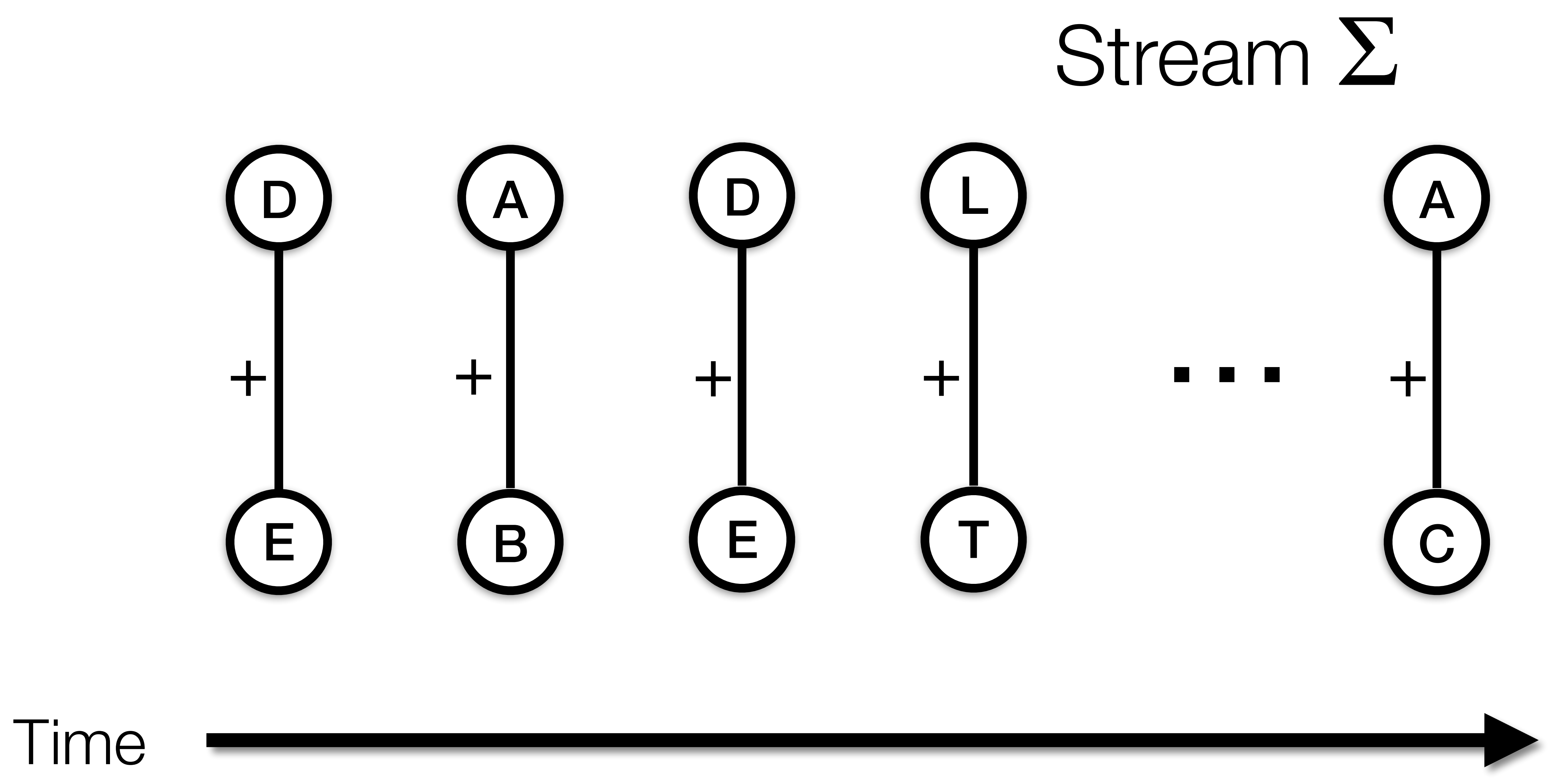
- Extensive experimental evaluation shows improvements and scalability of *Tonic* with respect to the state of the art

Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t:



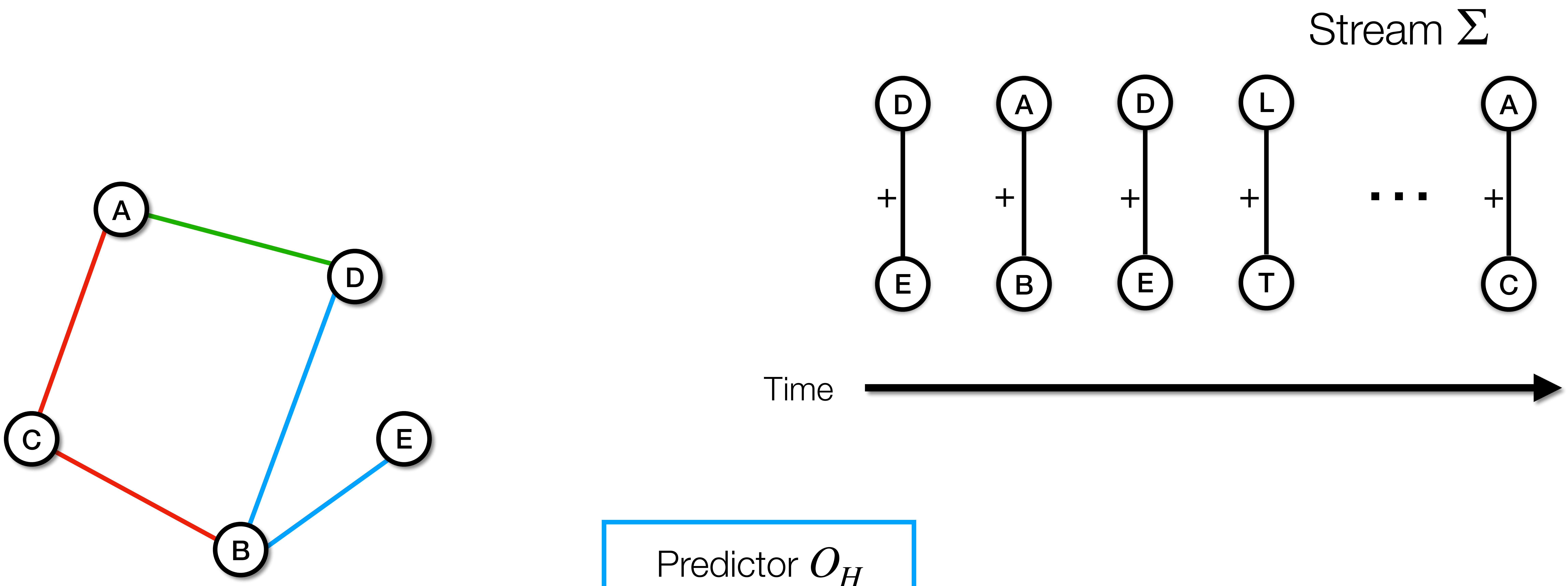
Current Sample



Predictor O_H

Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t:



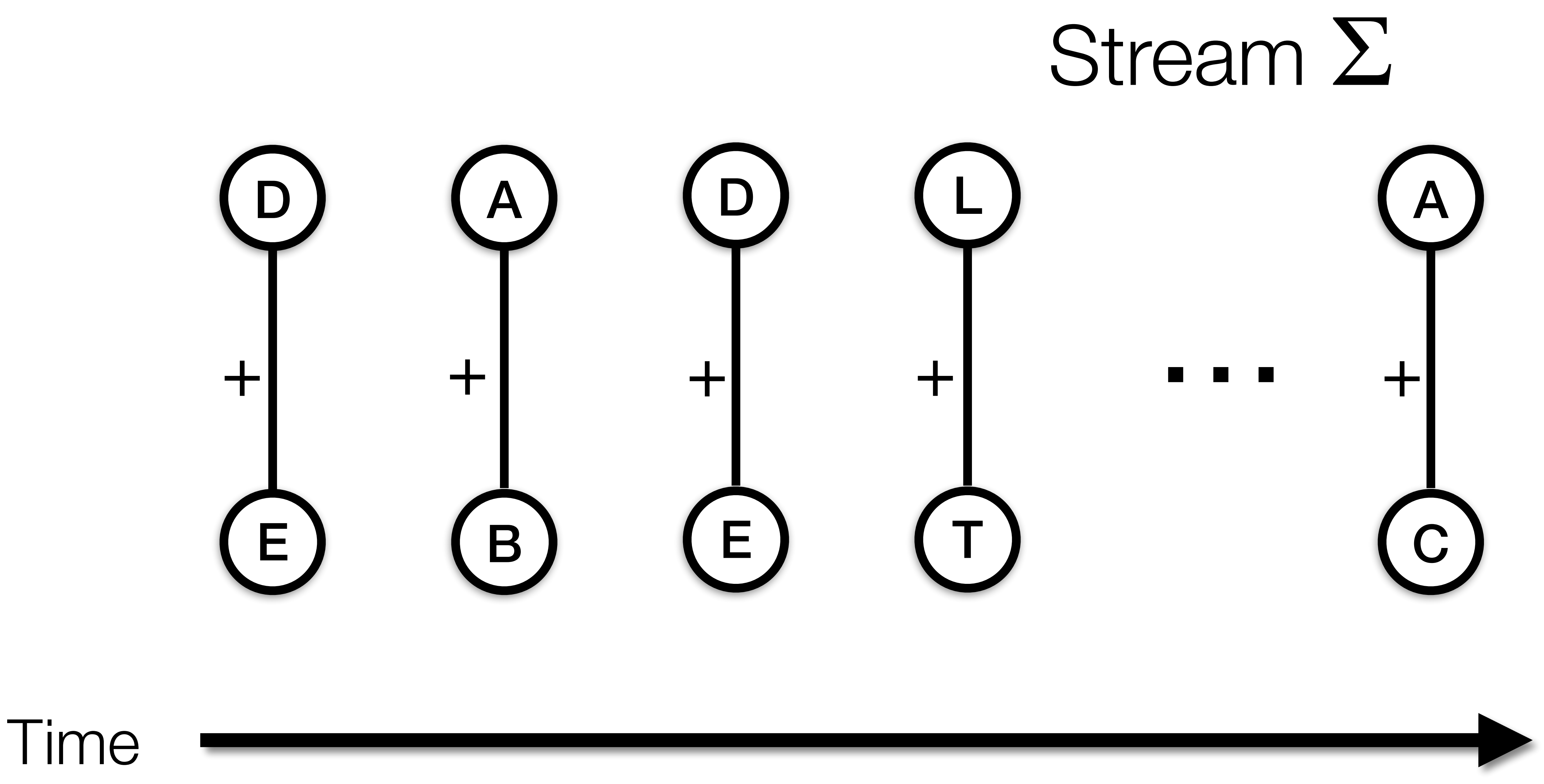
Predictor O_H

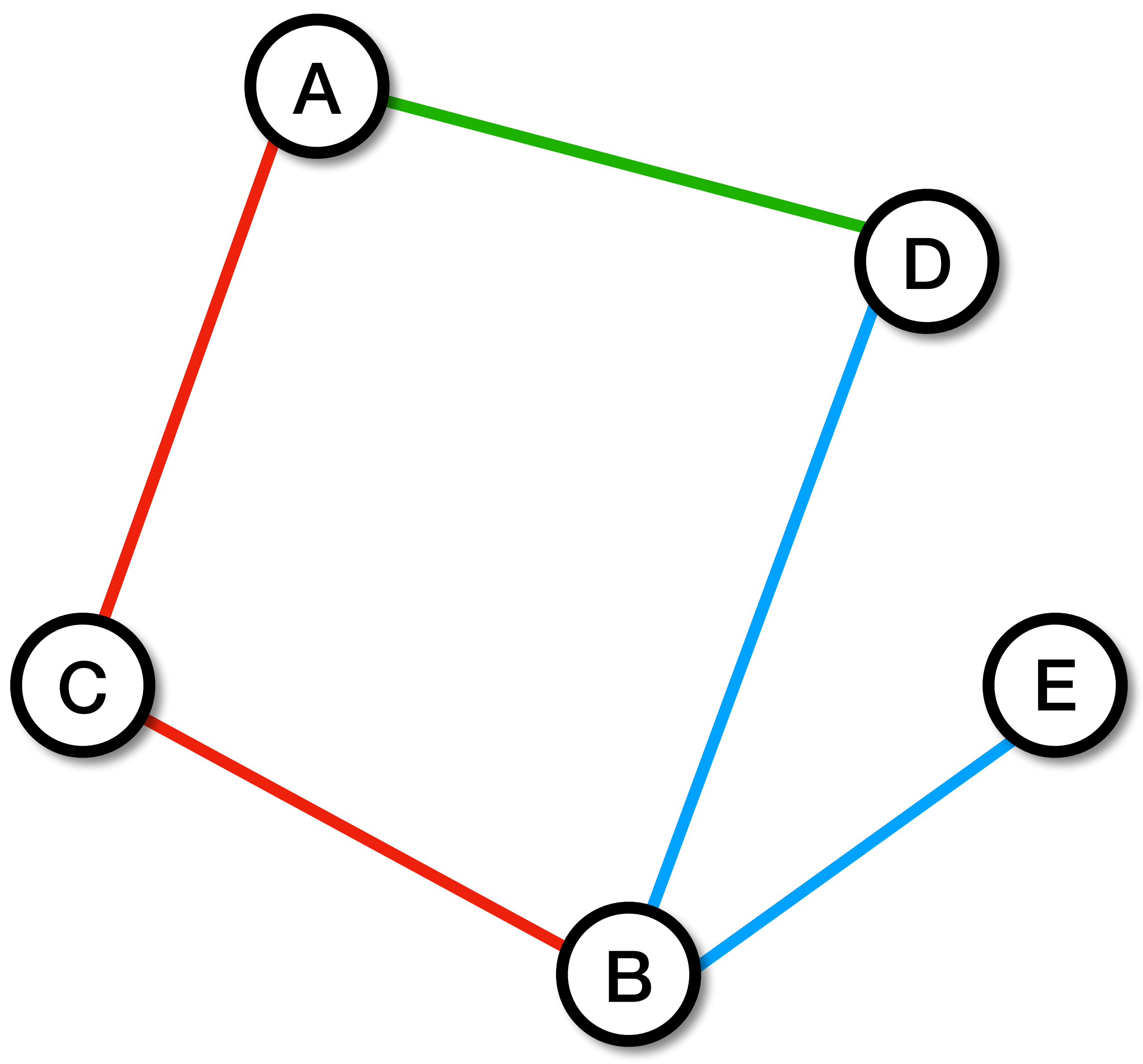
Current Sample $W^{(t)} \cup H^{(t)} \cup S^{(t)}$

Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t:

Count triangles closed by current edge $e^{(t)}$ in our sample.





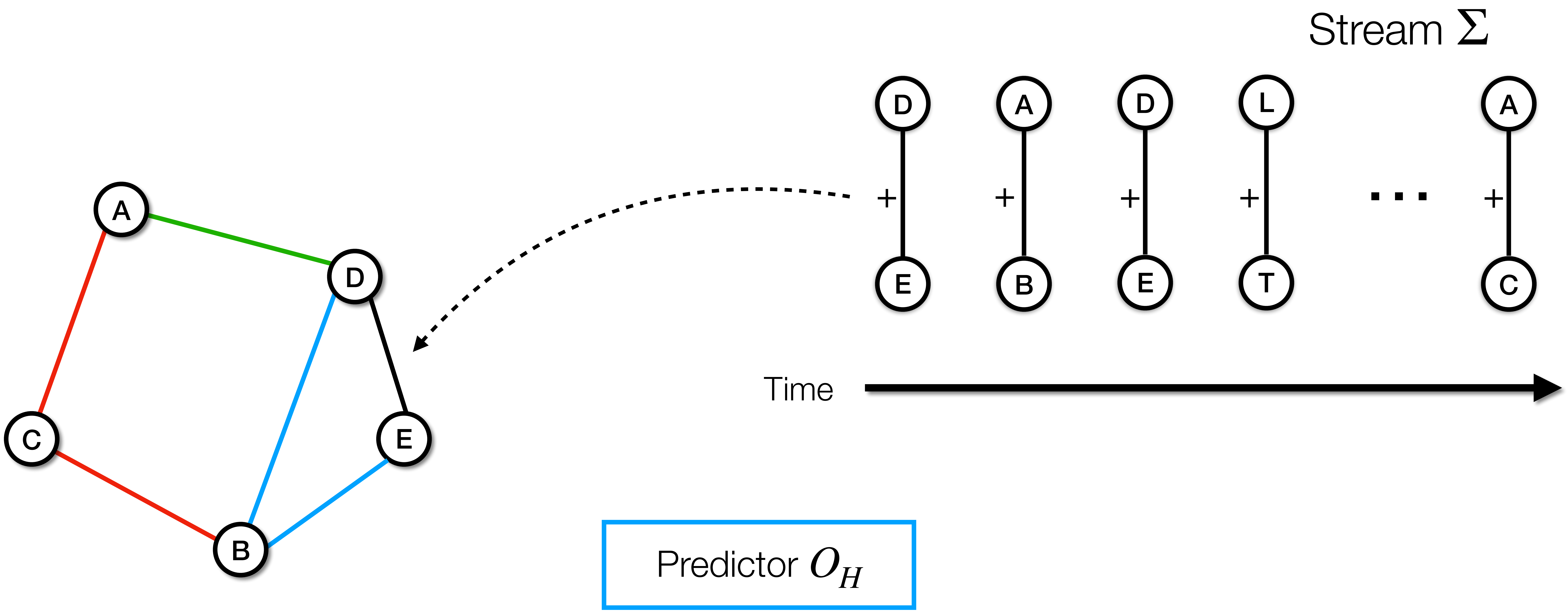
Current Sample $W^{(t)} \cup H^{(t)} \cup S^{(t)}$

Predictor O_H

Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t:

Count triangles closed by current edge $e^{(t)}$ in our sample.

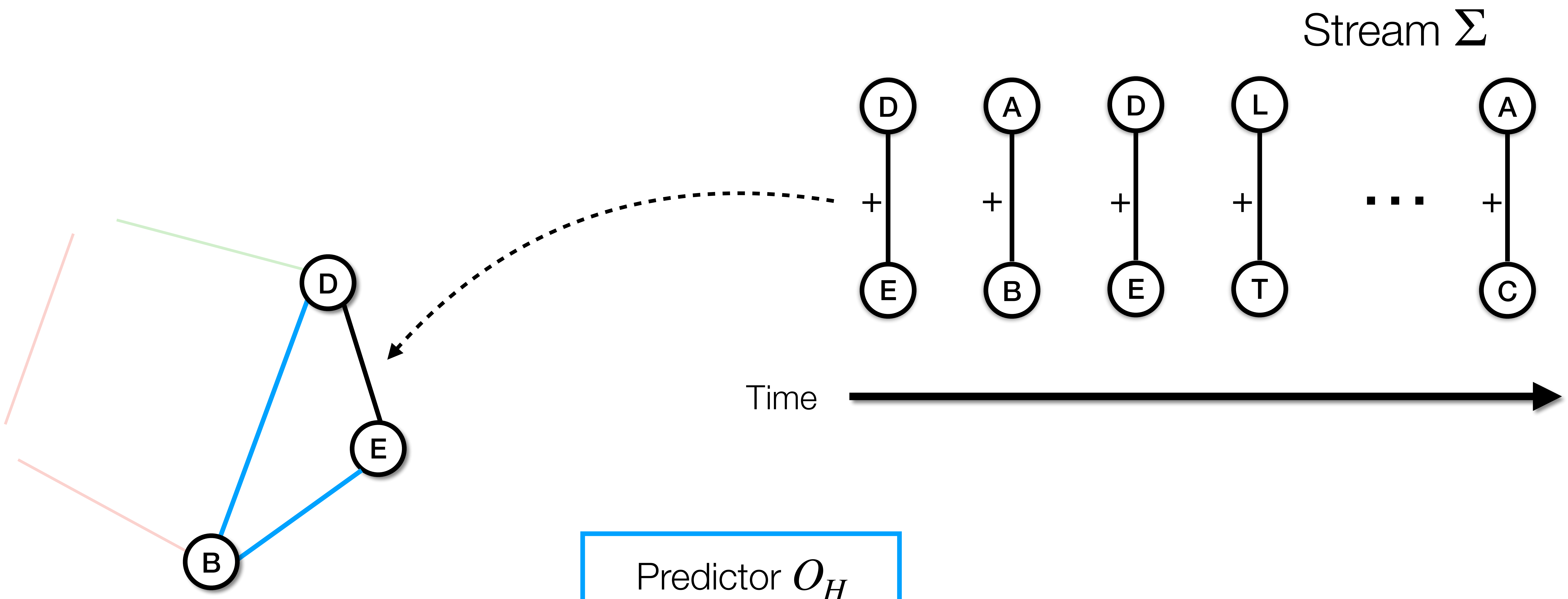


Current Sample $W^{(t)} \cup H^{(t)} \cup S^{(t)}$

Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t:

Count triangles closed by current edge $e^{(t)}$ in our sample.



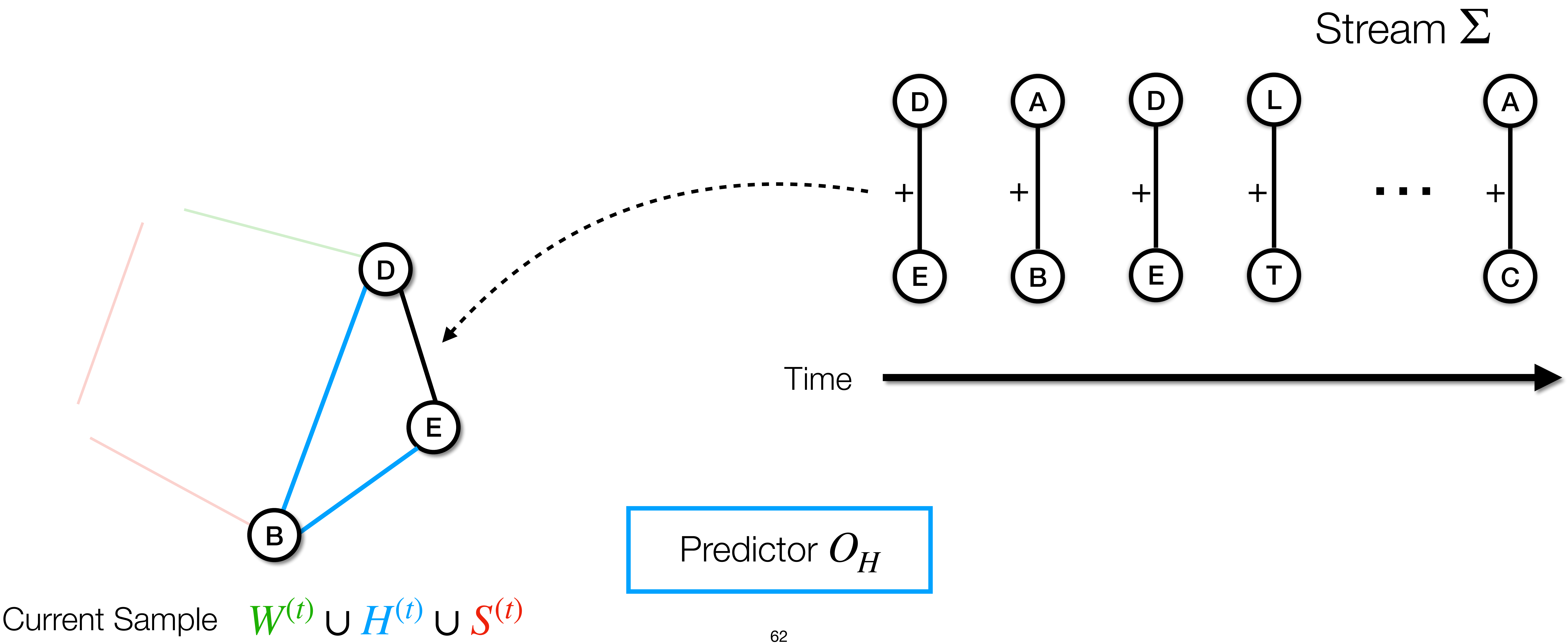
Predictor O_H

Current Sample $W^{(t)} \cup H^{(t)} \cup S^{(t)}$

Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t:

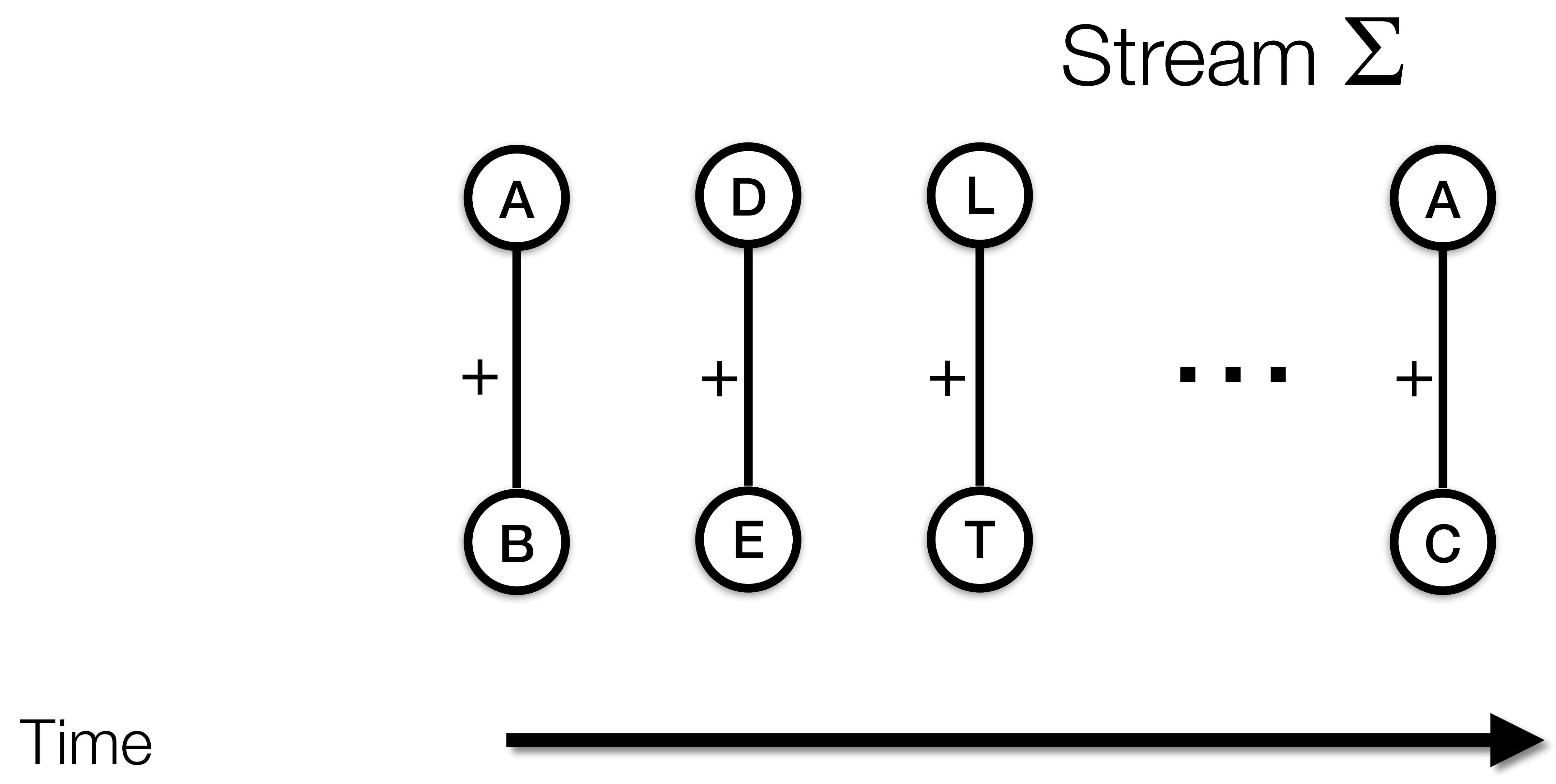
Triangles are weighted by the inverse probability with which edges have been sampled.

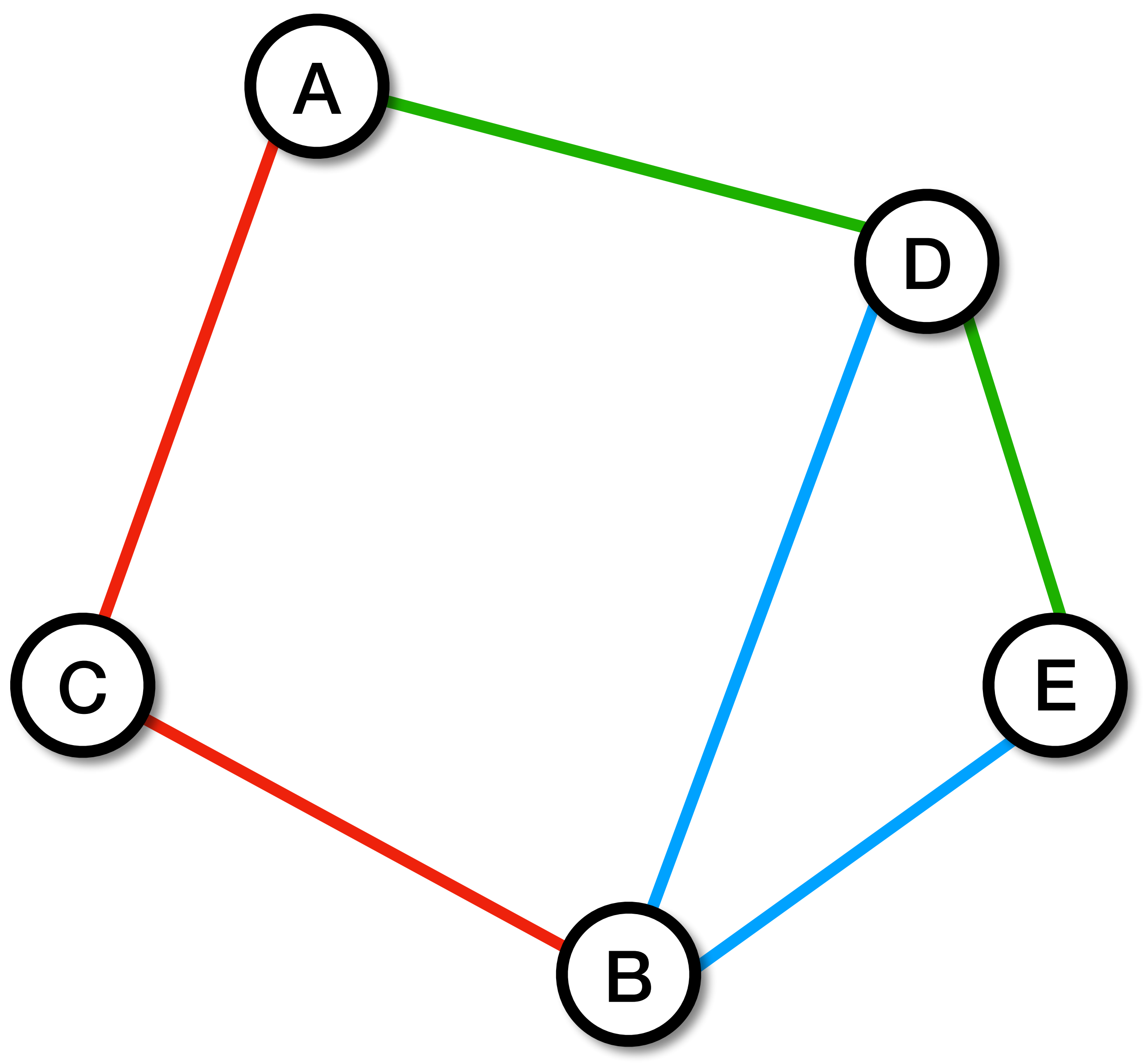


Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t:

Current edge $e^{(t)}$ is inserted in the waiting room.





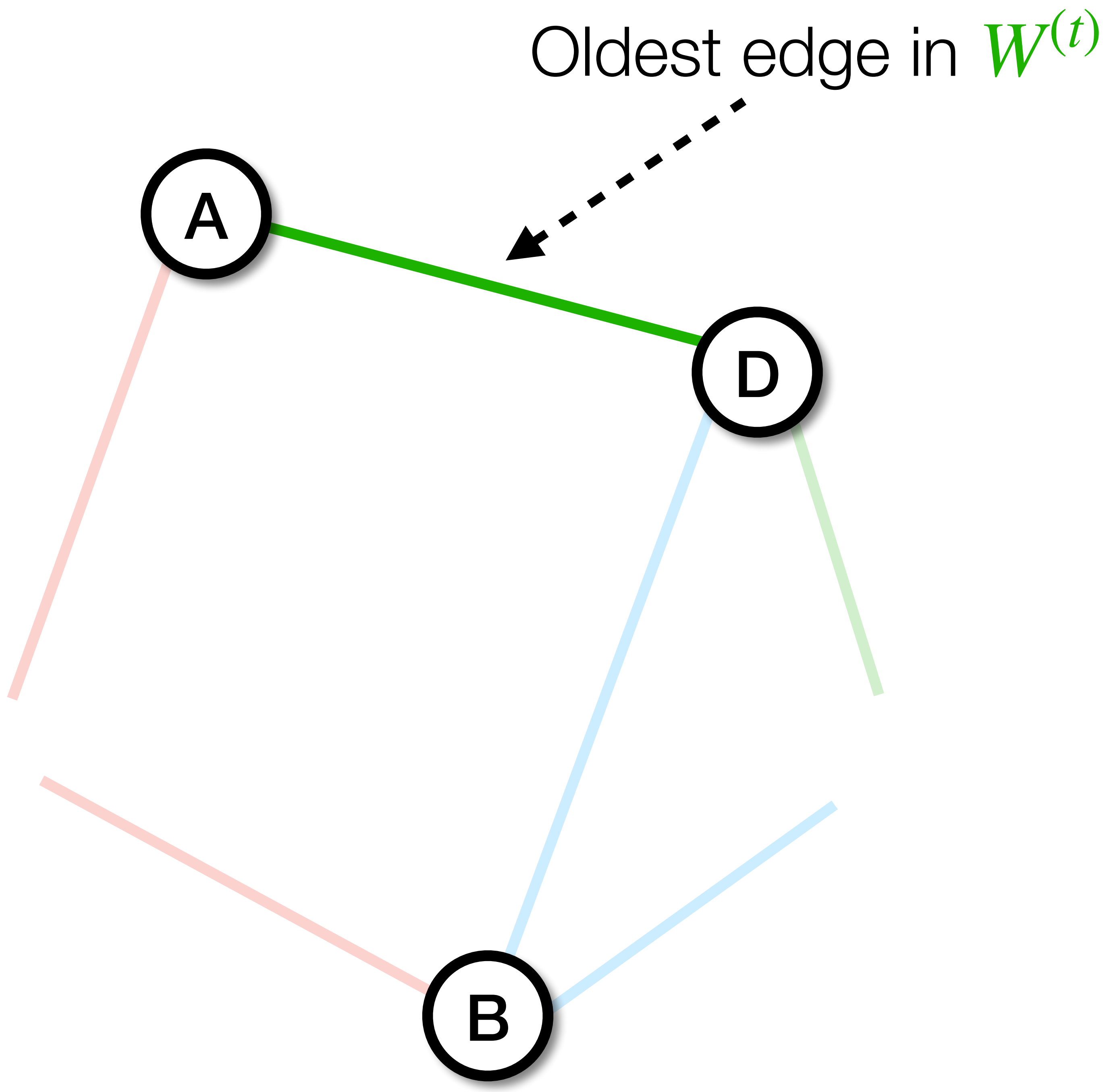
Current Sample $W^{(t)} \cup H^{(t)} \cup S^{(t)}$

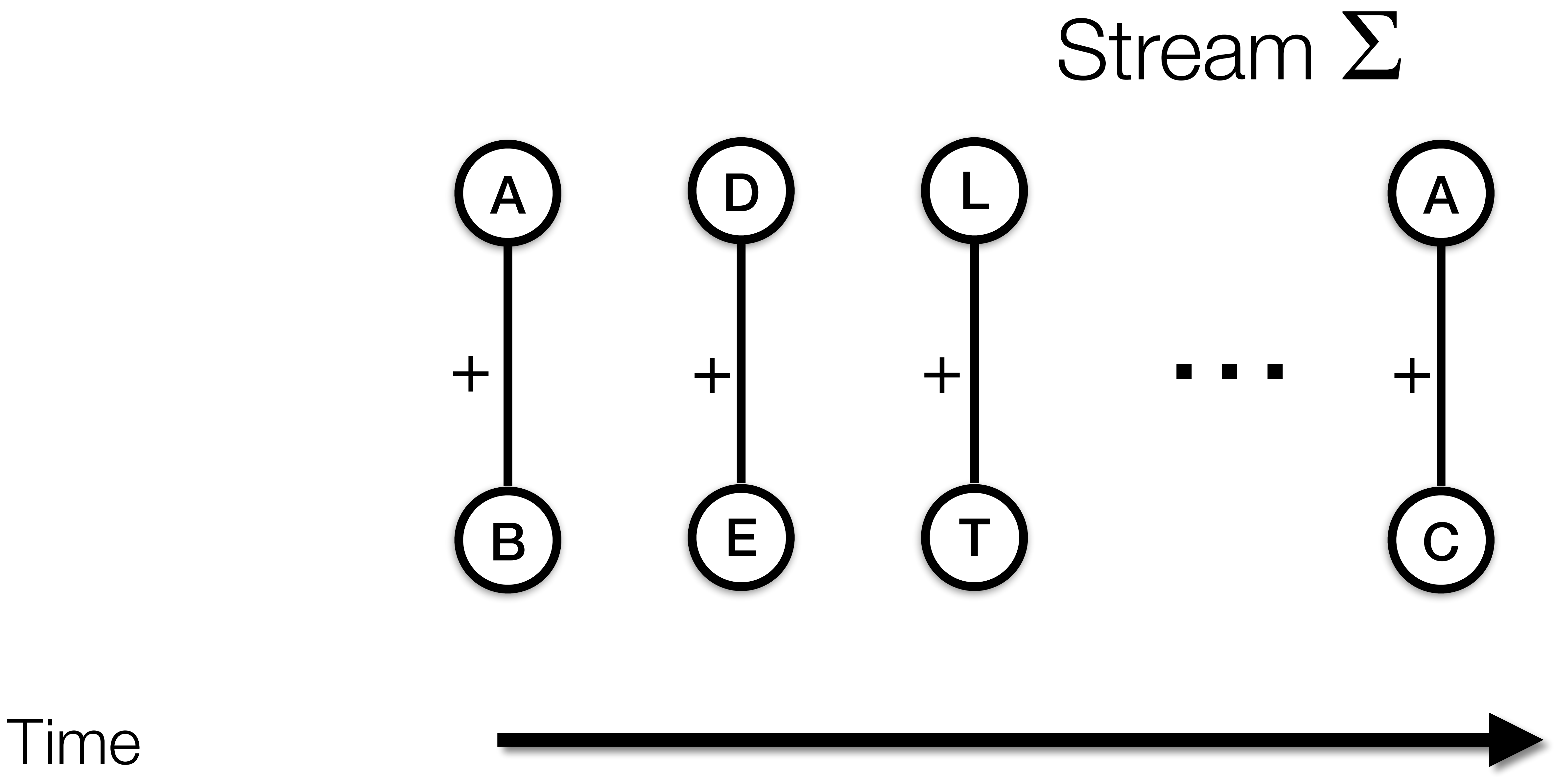
Predictor O_H

Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t:

If $W^{(t)}$ is full, pop oldest edge, and sample lightest (according to the predictor) between popped edge and edges in $H^{(t)}$.





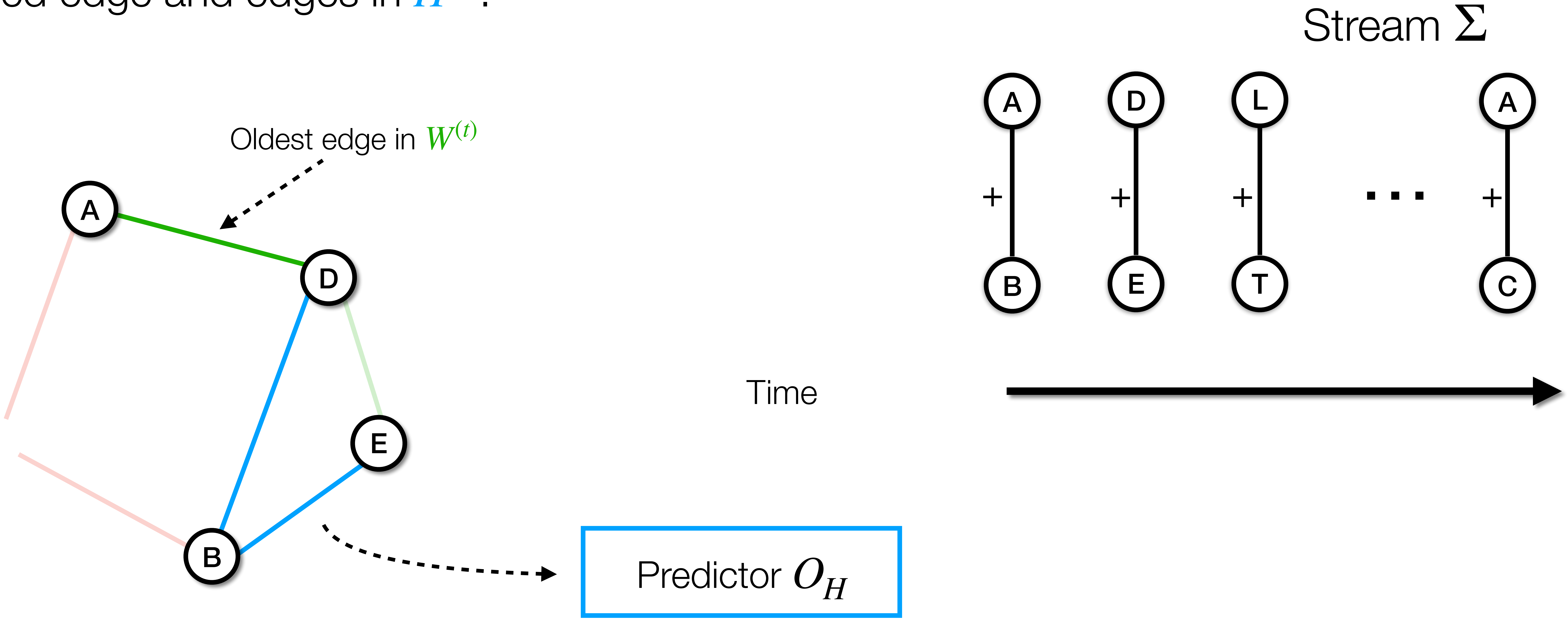
Predictor O_H

Current Sample $W^{(t)} \cup H^{(t)} \cup S^{(t)}$

Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t:

If $W^{(t)}$ is full, pop oldest edge, and sample lightest (according to the predictor) between popped edge and edges in $H^{(t)}$.

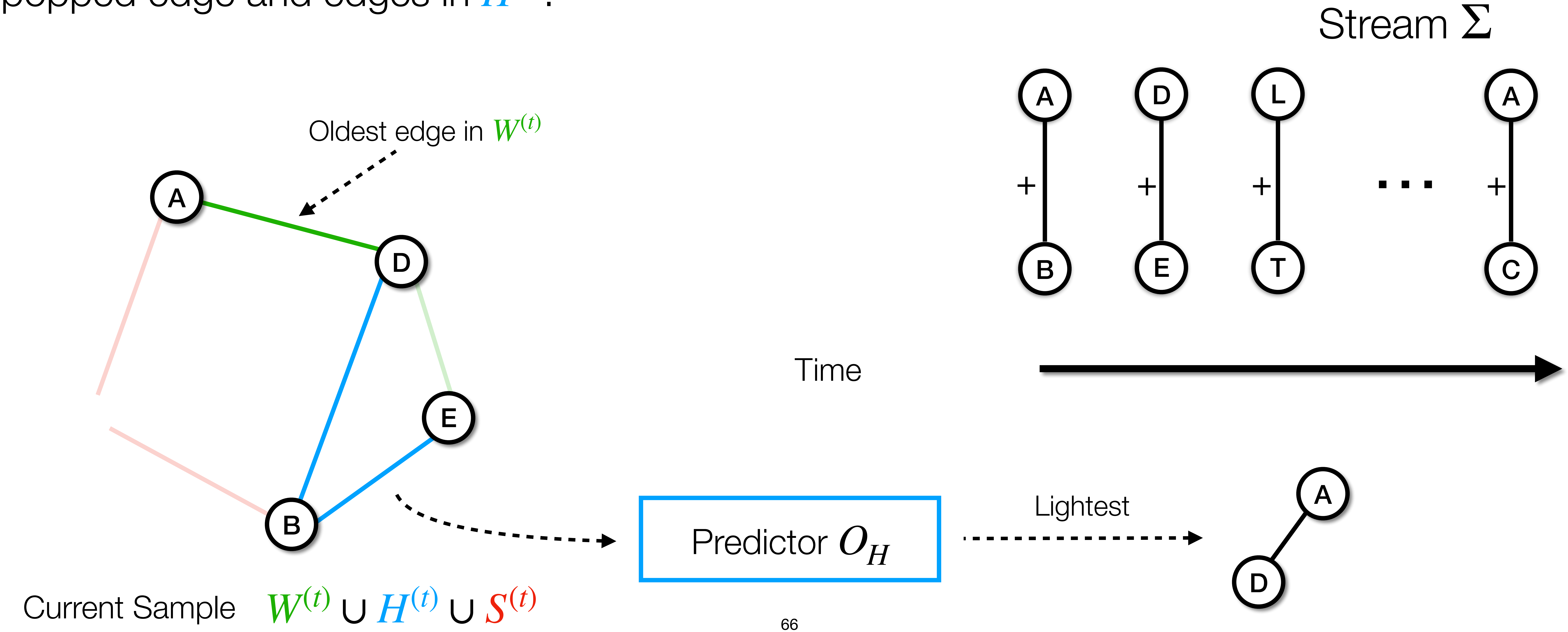


Current Sample $W^{(t)} \cup H^{(t)} \cup S^{(t)}$

Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t:

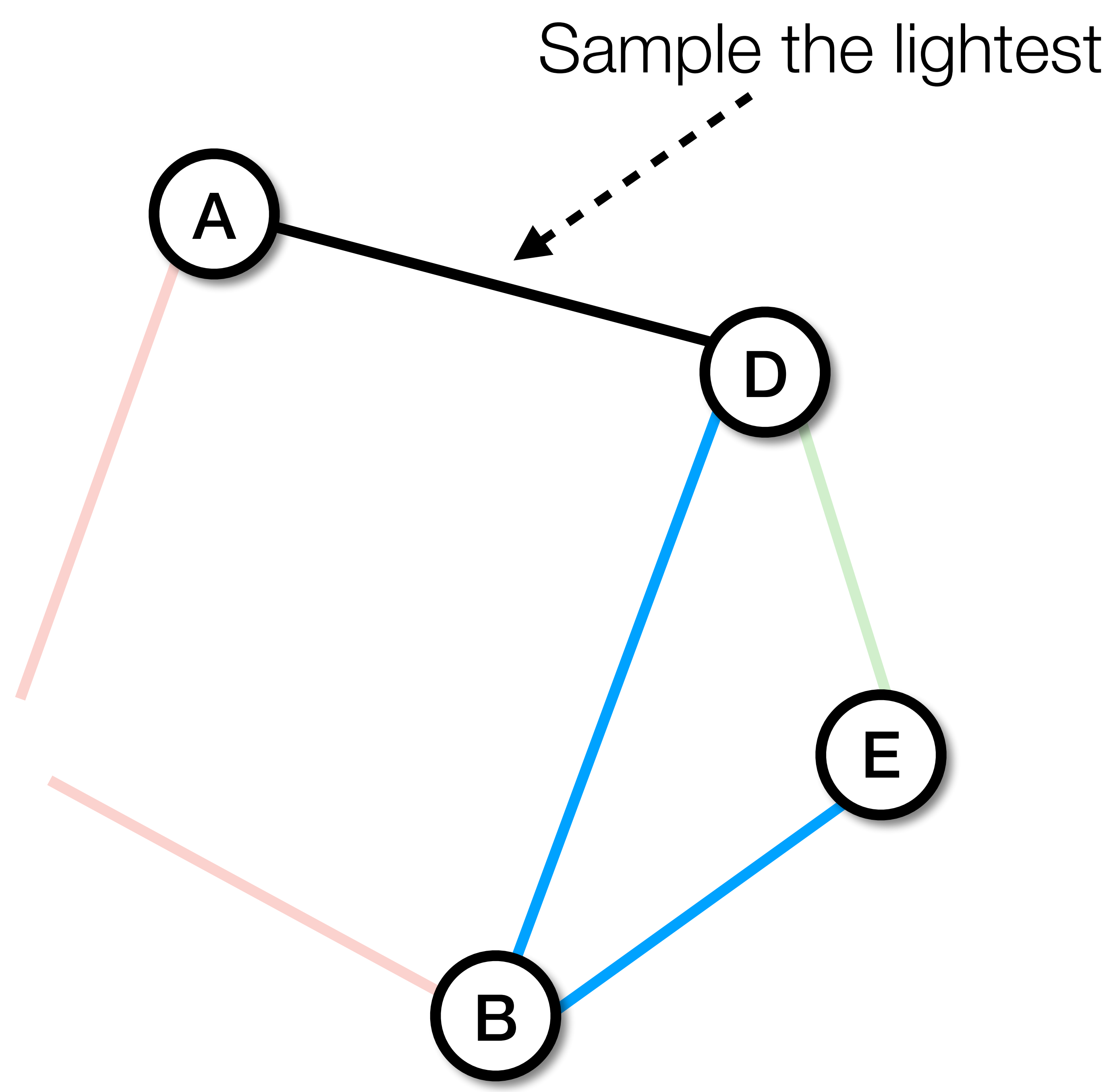
If $W^{(t)}$ is full, pop oldest edge, and sample lightest (according to the predictor) between popped edge and edges in $H^{(t)}$.



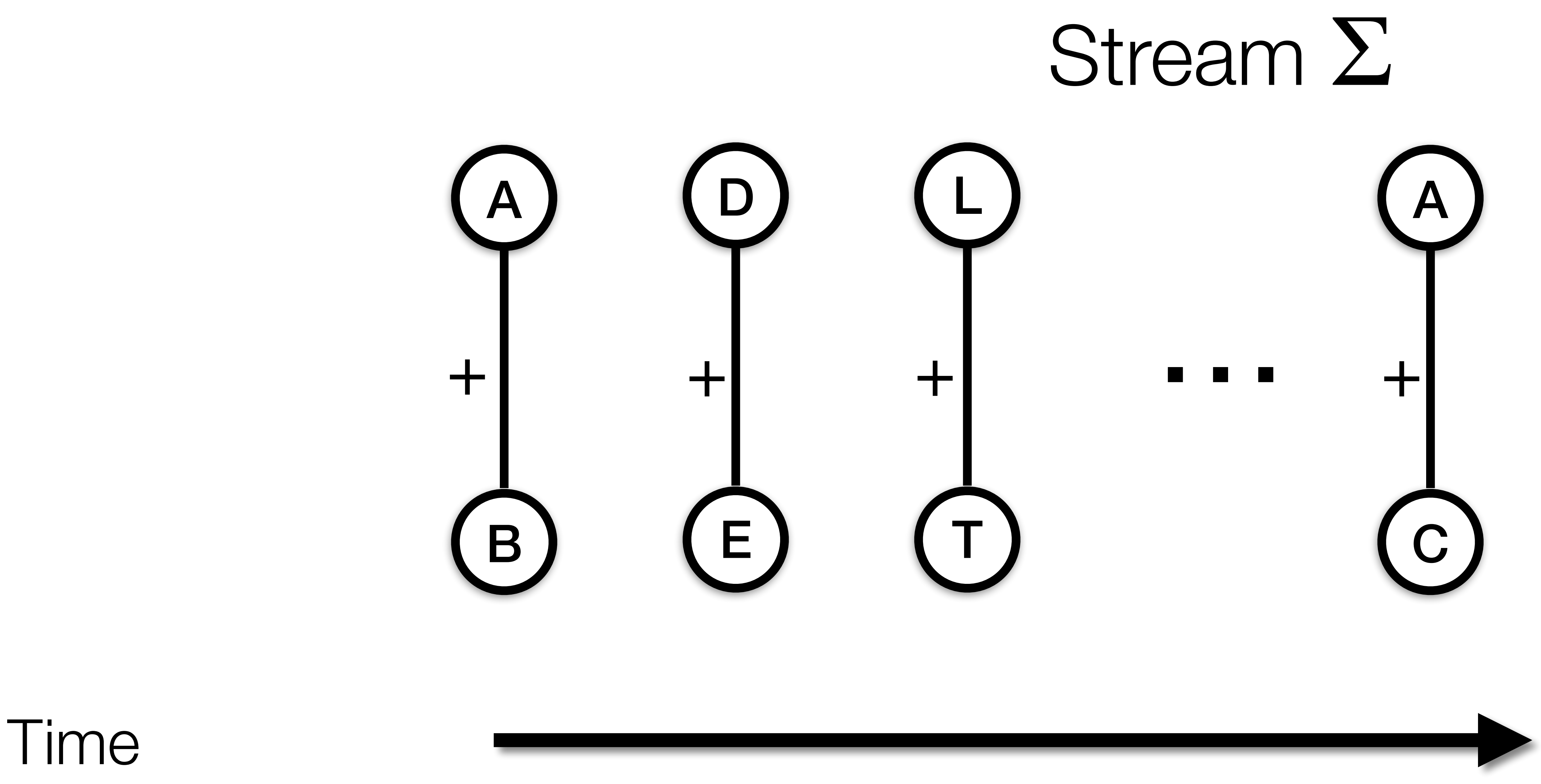
Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t:

If $W^{(t)}$ is full, pop oldest edge, and sample lightest (according to the predictor) between popped edge and edges in $H^{(t)}$.



Current Sample $W^{(t)} \cup H^{(t)} \cup S^{(t)}$

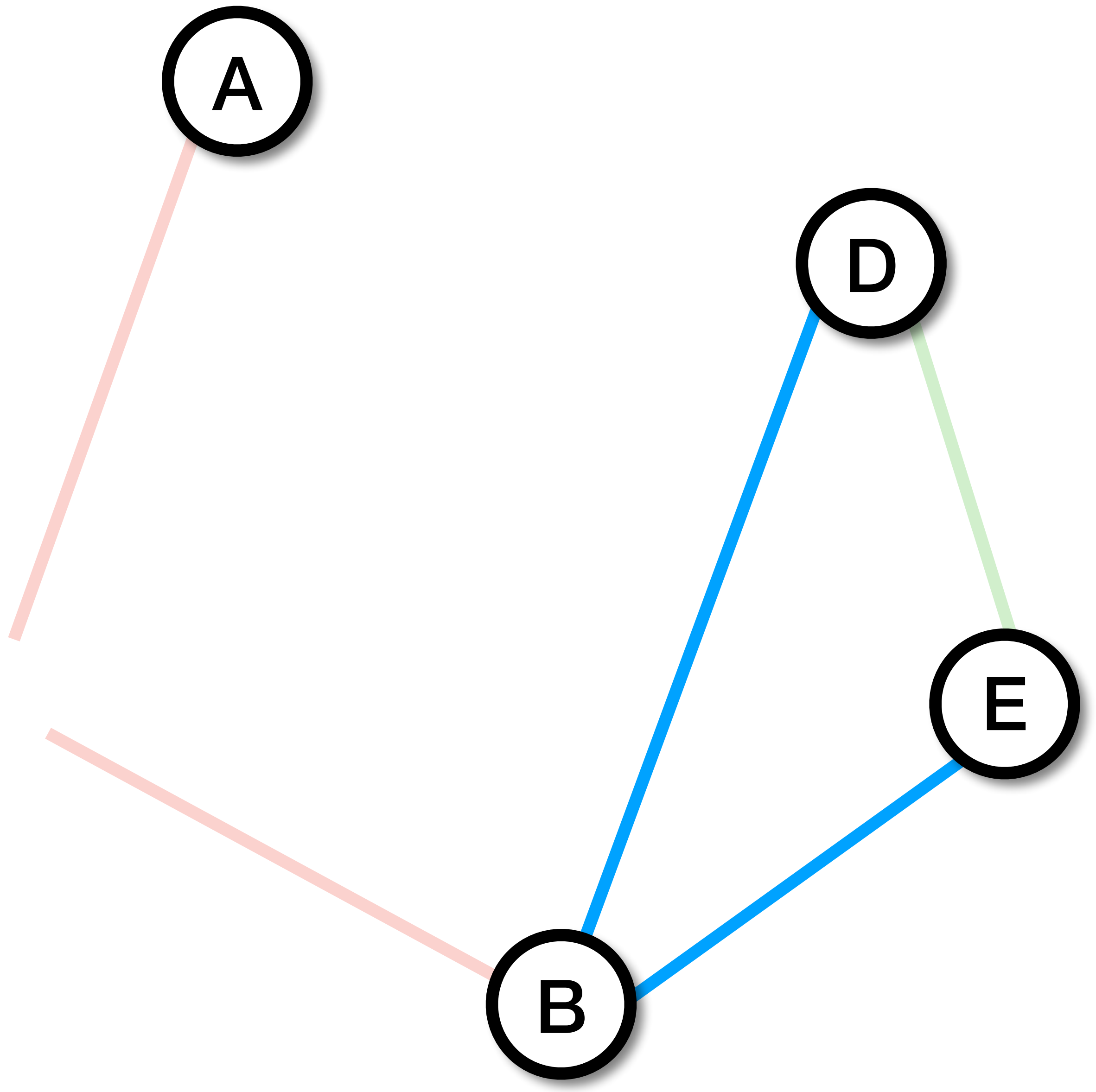


Predictor O_H

Tonic: Overall Algorithm

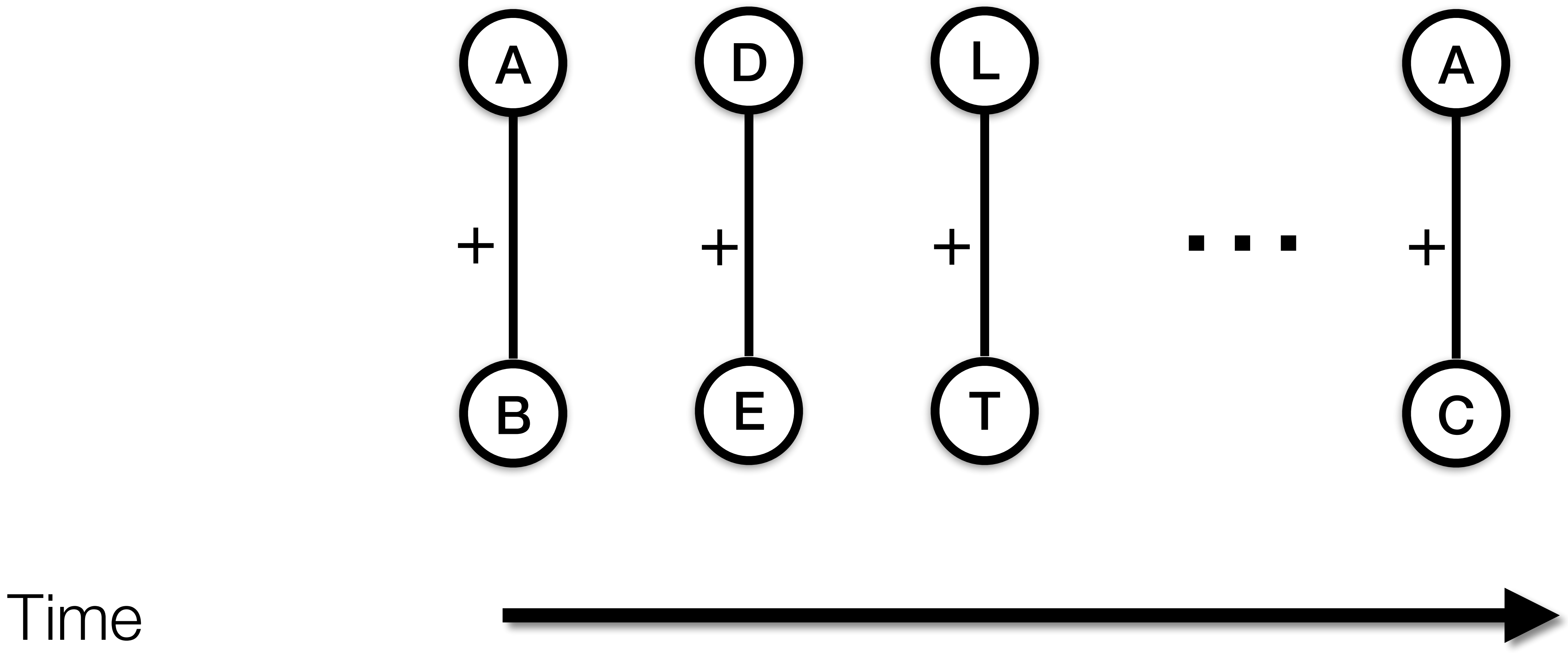
For each edge $e^{(t)}$ observed on the stream Σ at time t:

If $W^{(t)}$ is full, pop oldest edge, and sample lightest (according to the predictor) between popped edge and edges in $H^{(t)}$.



Current Sample $W^{(t)} \cup H^{(t)} \cup S^{(t)}$

Stream Σ



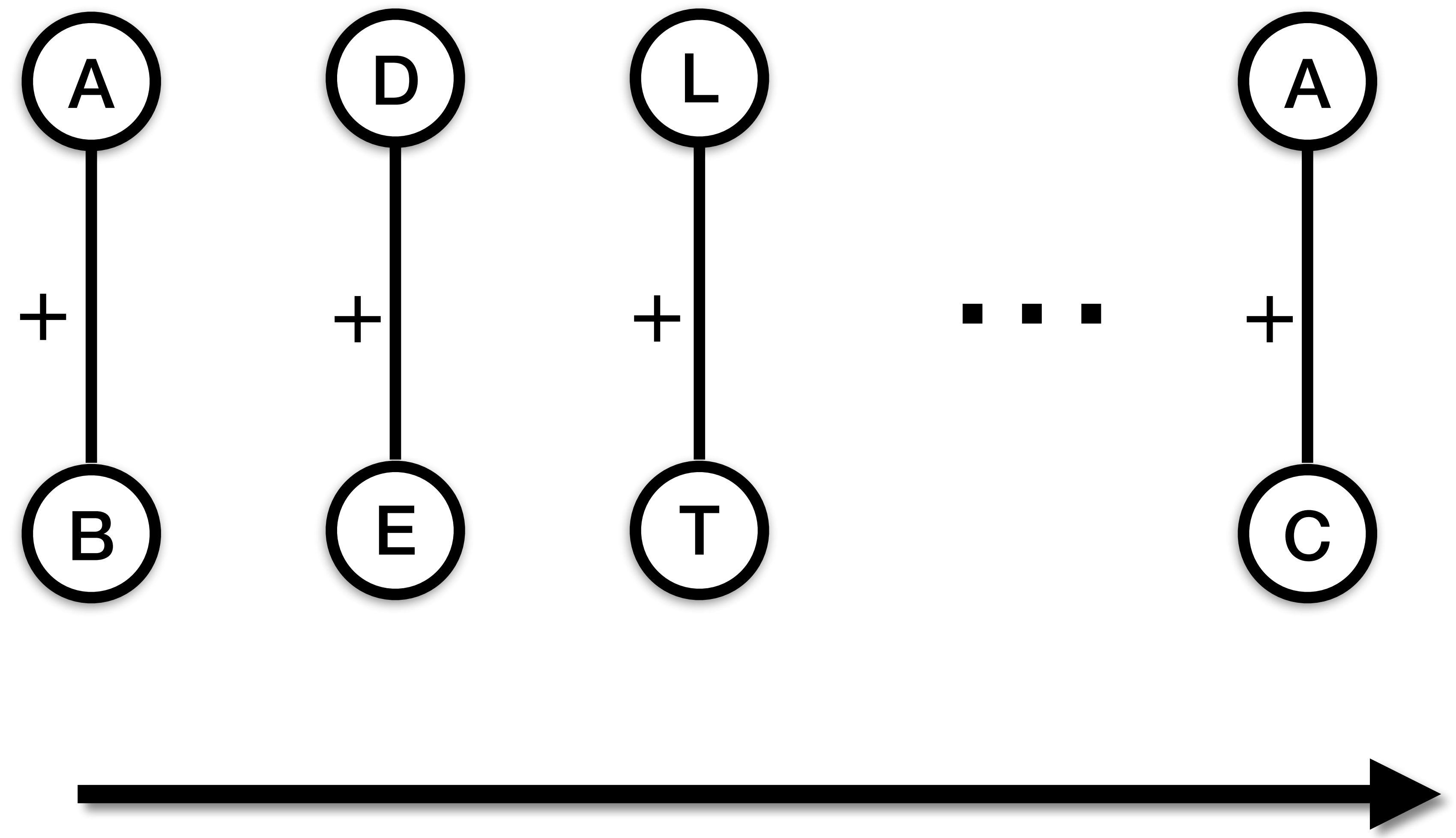
Predictor O_H

Tonic: Overall Algorithm

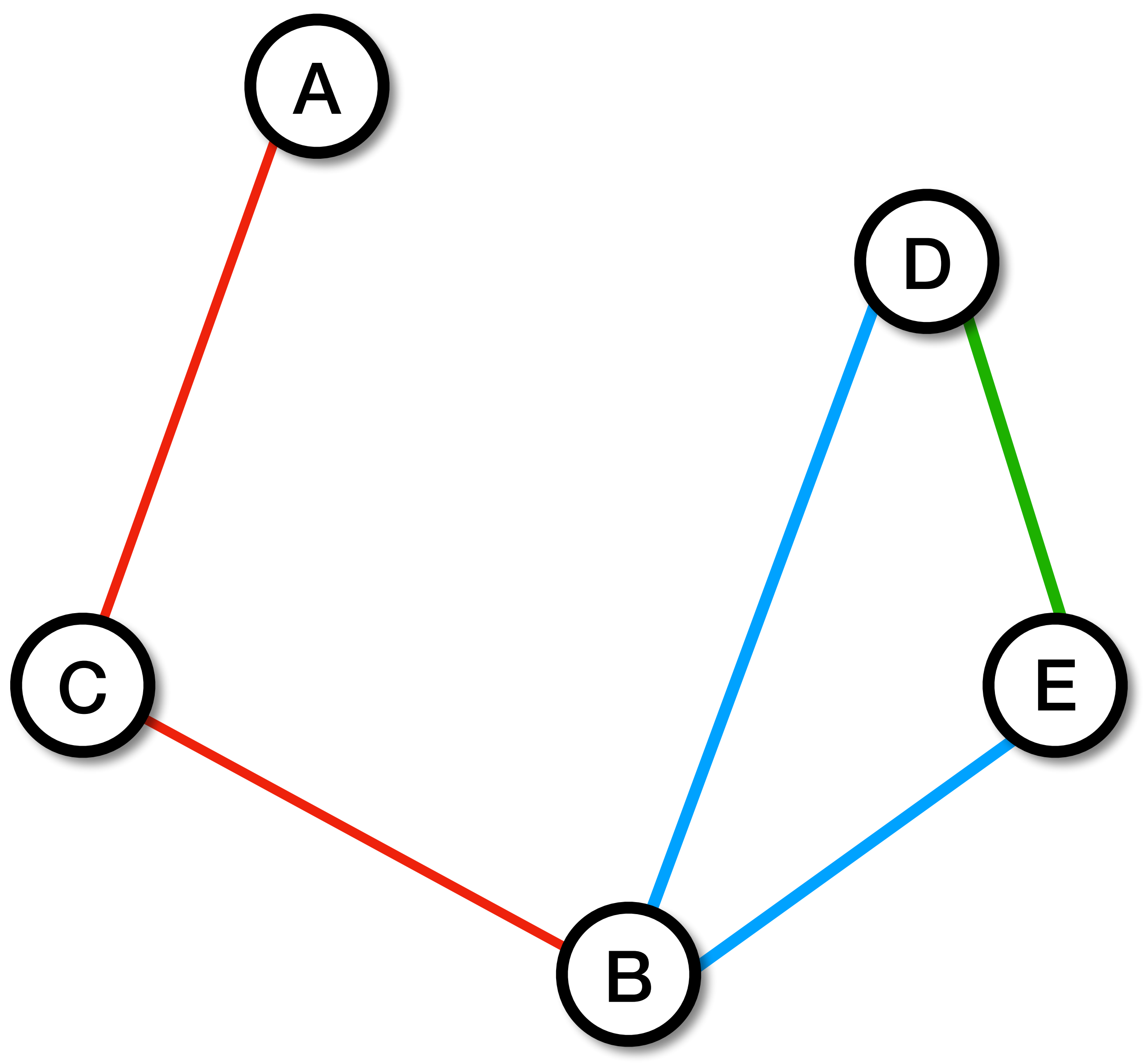
For each edge $e^{(t)}$ observed on the stream Σ at time t:

If $W^{(t)}$ is full, pop oldest edge, and sample lightest (according to the predictor) between popped edge and edges in $H^{(t)}$.

Stream Σ



Time



Current Sample $W^{(t)} \cup H^{(t)} \cup S^{(t)}$

Predictor O_H

A Practical Heaviness Predictor

We do not make any assumption on the predictor used by *Tonic*.

A Practical Heaviness Predictor

We do not make any assumption on the predictor used by *Tonic*.

We propose a simple, practical and application-independent predictor:

MinDegreePredictor stores $\bar{n}$ highest-degree **nodes** and **degrees**. Given edge $e = \{u, v\}$, outputs: $O_H\left(\{u, v\}\right) = \min\left(deg(u), deg(v)\right)$ if both u and v are present, 0 otherwise.

MinDegreePredictor

$\bar{n}$

v_1	$deg(v_1)$
u_1	$deg(u_1)$
v_2	$deg(v_2)$
$\vdots$	$\vdots$
v_3	$deg(v_3)$

Tonic: theoretical analysis

Theorem (Unbiasedness of estimates): let $T^{(t)}$ be the true number of global triangles. Then, *Tonic* outputs $\hat{T}^{(t)}$ such that:

$$\mathbb{E}\left[\hat{T}^{(t)}\right] = T^{(t)}, \forall\ t \geq 0$$

Tonic: theoretical analysis

We prove that *useful* predictions in *Tonic* leads to better estimates than using *WRS* alone.

Tonic: theoretical analysis

We prove that *useful* predictions in *Tonic* leads to better estimates than using *WRS* alone.

Consider:

- *WRS* sampling edges leaving the waiting room with probability p
- *Tonic* sampling light edges with probability $p' < p$
- We define an edge e as **heavy** if e appears in $\geq \rho$ triangles (otherwise, **light**)
- **Errors of predictions:** heavy edges involved in $\geq \rho \cdot c$ triangles, light edges involved in $\leq \rho / c$ triangles, for some $c \geq 1$. For edges with heaviness $\in \left[\rho / c \ , \rho \cdot c\right]$, the predictor can make arbitrarily wrong choices

Tonic: theoretical analysis

Let T_H the total number of triangles in which **heavy** edges appear, and T_L similarly for **light** edges.

Tonic: theoretical analysis

Let T_H the total number of triangles in which **heavy** edges appear, and T_L similarly for **light** edges.

Proposition (informal): the variance of *Tonic* estimates is less than the variance of *WRS* estimates if:

$$T_H > 3 \, \frac{(1/p'^2 - 1/p^2) + c\rho(1/p' - 1/p)}{(1/p - 1)(3 + 4\rho/c)} \cdot T_L$$

Tonic: theoretical analysis

Let T_H the total number of triangles in which **heavy** edges appear, and T_L similarly for **light** edges.

Proposition (informal): the variance of *Tonic* estimates is less than the variance of *WRS* estimates if:

$$T_H > 3 \; \frac{(1/p'^2 - 1/p^2) + c\rho(1/p' - 1/p)}{(1/p - 1)(3 + 4\rho/c)} \cdot T_L$$

Interpretation: *useful* predictions (predicted *heavy* edges are involved in a sufficient number of triangles), lead to better estimates.

Experimental Evaluation

We consider real-world **single graph streams**, from social network, citation network.

Compare *Tonic* with state-of-the-art: algorithms provided with same memory budget k.

TABLE I

DATASETS' STATISTICS: NUMBER n OF NODES; NUMBER m OF EDGES; NUMBER T OF TRIANGLES

Dataset	n	m	T
Single Graphs			
Edit EN Wikibooks	$133k$	$386k$	$178k$
SOC YouTube Growth	$3.2M$	$9.3M$	$12.3M$
Cit US Patents	$3.7M$	$16.5M$	$7.5M$
Actors Collaborations	$382k$	$15M$	$346.8M$
Stackoverflow	$2.5M$	$28.1M$	$114.2M$
SOC LiveJournal	$4.8M$	$42.8M$	$285.7M$
Twitter-merged	$41M$	$1.2B$	$34.8B$

Experimental Evaluation

We consider real-world **single graph streams**, from social network, citation network.

Compare *Tonic* with state-of-the-art: algorithms provided with same memory budget k.

TABLE I
DATASETS' STATISTICS: NUMBER n OF NODES; NUMBER m OF EDGES; NUMBER T OF TRIANGLES

Dataset	n	m	T
Single Graphs			
Edit EN Wikibooks	$133k$	$386k$	$178k$
SOC YouTube Growth	$3.2M$	$9.3M$	$12.3M$
Cit US Patents	$3.7M$	$16.5M$	$7.5M$
Actors Collaborations	$382k$	$15M$	$346.8M$
Stackoverflow	$2.5M$	$28.1M$	$114.2M$
SOC LiveJournal	$4.8M$	$42.8M$	$285.7M$
Twitter-merged	$41M$	$1.2B$	$34.8B$

Global Relative Error:

$$|\hat{T} - T| \ / \ T$$

Edge Heaviness Predictors

In our experiments we considered:

- *OracleExact,* stores the value of $\Delta(e)$ for top 10% $(m/10)$ **heaviest edges** e

OracleExact

u_1	v_1	$\Delta\left(\{u_1, v_1\}\right)$
u_2	v_1	$\Delta\left(\{u_2, v_1\}\right)$
u_3	v_3	$\Delta\left(\{u_3, v_3\}\right)$
u_2	v_4	$\Delta\left(\{u_4, v_4\}\right)$
$\vdots$	$\vdots$	$\vdots$
u_1	v_5	$\Delta\left(\{u_1, v_5\}\right)$

Edge Heaviness Predictors

In our experiments we considered:

- *OracleExact*
- *Oracle-noWR*, subtracts to $\Delta(e)$ the triangles for which e is in the **waiting room**, for top 10% edges

OracleExact

u_1	v_1	$\Delta(\{u_1, v_1\})$
u_2	v_1	$\Delta(\{u_2, v_1\})$
u_3	v_3	$\Delta(\{u_3, v_3\})$
u_2	v_4	$\Delta(\{u_4, v_4\})$
$\vdots$	$\vdots$	$\vdots$
u_1	v_5	$\Delta(\{u_1, v_5\})$

Oracle-noWR

u_2	v_1	$\Delta'(\{u_2, v_1\})$
u_3	v_3	$\Delta'(\{u_3, v_3\})$
u_2	v_4	$\Delta'(\{u_2, v_4\})$
u_7	v_7	$\Delta'(\{u_7, v_7\})$
$\vdots$	$\vdots$	$\vdots$
u_1	v_4	$\Delta'(\{u_1, v_4\})$

Edge Heaviness Predictors

In our experiments we considered:

- *OracleExact*
- *Oracle-noWR*, subtracts to $\Delta(e)$ the triangles for which e is in the **waiting room**, for top 10% edges

OracleExact

u_1	v_1	$\Delta(\{u_1, v_1\})$
u_2	v_1	$\Delta(\{u_2, v_1\})$
u_3	v_3	$\Delta(\{u_3, v_3\})$
u_2	v_4	$\Delta(\{u_4, v_4\})$
$\vdots$	$\vdots$	$\vdots$
u_1	v_5	$\Delta(\{u_1, v_5\})$

Oracle-noWR

u_2	v_1	$\Delta'(\{u_2, v_1\})$
u_3	v_3	$\Delta'(\{u_3, v_3\})$
u_2	v_4	$\Delta'(\{u_2, v_4\})$
u_7	v_7	$\Delta'(\{u_7, v_7\})$
$\vdots$	$\vdots$	$\vdots$
u_1	v_4	$\Delta'(\{u_1, v_4\})$

Impractical Predictors

Edge Heaviness Predictors

In our experiments we considered:

- *OracleExact*
- *Oracle-noWR*
- *MinDegreePredictor*, stores $\bar{n}$ highest-degree **nodes** and **degrees**. Given edge $e = \{u, v\}$, outputs: $O_H\left(\{u, v\}\right) = \min\left(deg(u), deg(v)\right)$

OracleExact

u_1	v_1	$\Delta\left(\{u_1, v_1\}\right)$
u_2	v_1	$\Delta\left(\{u_2, v_1\}\right)$
u_3	v_3	$\Delta\left(\{u_3, v_3\}\right)$
u_2	v_4	$\Delta\left(\{u_4, v_4\}\right)$
$\vdots$	$\vdots$	$\vdots$
u_1	v_5	$\Delta\left(\{u_1, v_5\}\right)$

Oracle-noWR

u_2	v_1	$\Delta'\left(\{u_2, v_1\}\right)$
u_3	v_3	$\Delta'\left(\{u_3, v_3\}\right)$
u_2	v_4	$\Delta'\left(\{u_2, v_4\}\right)$
u_7	v_7	$\Delta'\left(\{u_7, v_7\}\right)$
$\vdots$	$\vdots$	$\vdots$
u_1	v_4	$\Delta'\left(\{u_1, v_4\}\right)$

MinDegreePredictor

v_1	$deg(v_1)$
u_1	$deg(u_1)$
v_2	$deg(v_2)$
$\vdots$	$\vdots$
v_3	$deg(v_3)$

Edge Heaviness Predictors

In our experiments we considered:

- *OracleExact*
- *Oracle-noWR*
- *MinDegreePredictor*

In practice, $\bar{n} \ll m/10$!

OracleExact

$\frac{m}{10}$

u_1	v_1	$\Delta(\{u_1, v_1\})$
u_2	v_1	$\Delta(\{u_2, v_1\})$
u_3	v_3	$\Delta(\{u_3, v_3\})$
u_2	v_4	$\Delta(\{u_4, v_4\})$
$\vdots$	$\vdots$	$\vdots$
u_1	v_5	$\Delta(\{u_1, v_5\})$

Oracle-noWR

u_2	v_1	$\Delta'(\{u_2, v_1\})$
u_3	v_3	$\Delta'(\{u_3, v_3\})$
u_2	v_4	$\Delta'(\{u_2, v_4\})$
u_7	v_7	$\Delta'(\{u_7, v_7\})$
$\vdots$	$\vdots$	$\vdots$
u_1	v_4	$\Delta'(\{u_1, v_4\})$

MinDegreePredictor

$\bar{n}$

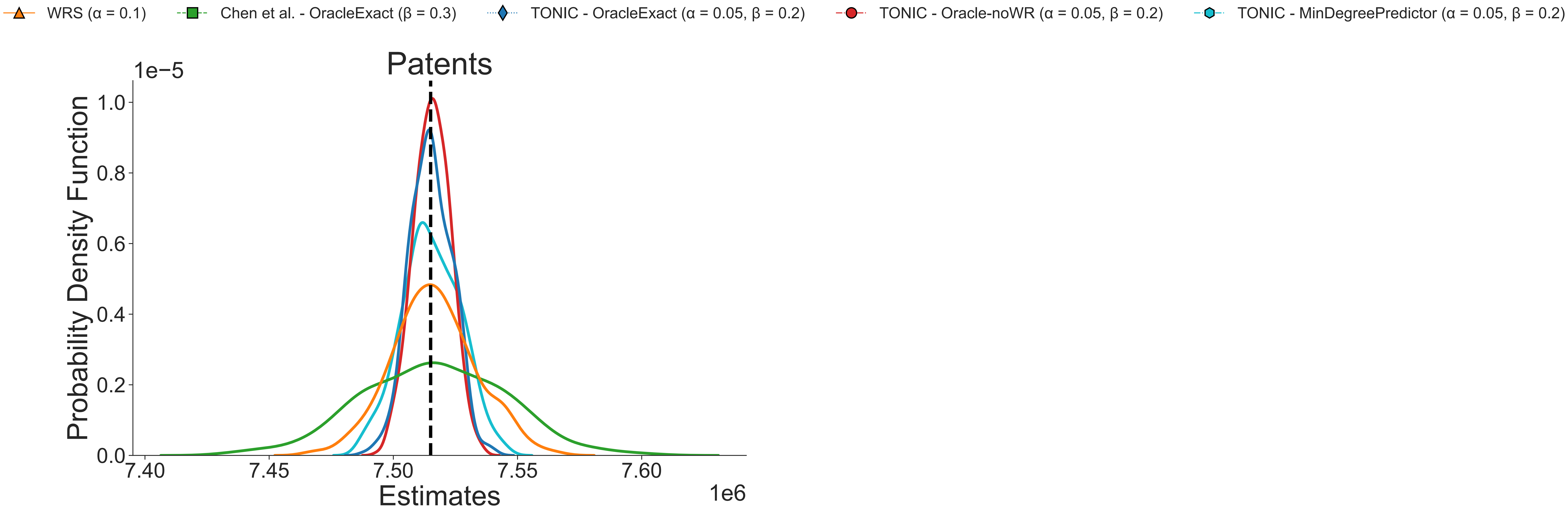
v_1	$deg(v_1)$
u_1	$deg(u_1)$
v_2	$deg(v_2)$
$\vdots$	$\vdots$
v_3	$deg(v_3)$

Experimental Evaluation

WRS (α = 0.1) Chen et al. - OracleExact (β = 0.3) TONIC - OracleExact (α = 0.05, β = 0.2) TONIC - Oracle-noWR (α = 0.05, β = 0.2) TONIC - MinDegreePredictor (α = 0.05, β = 0.2)

Experimental Evaluation

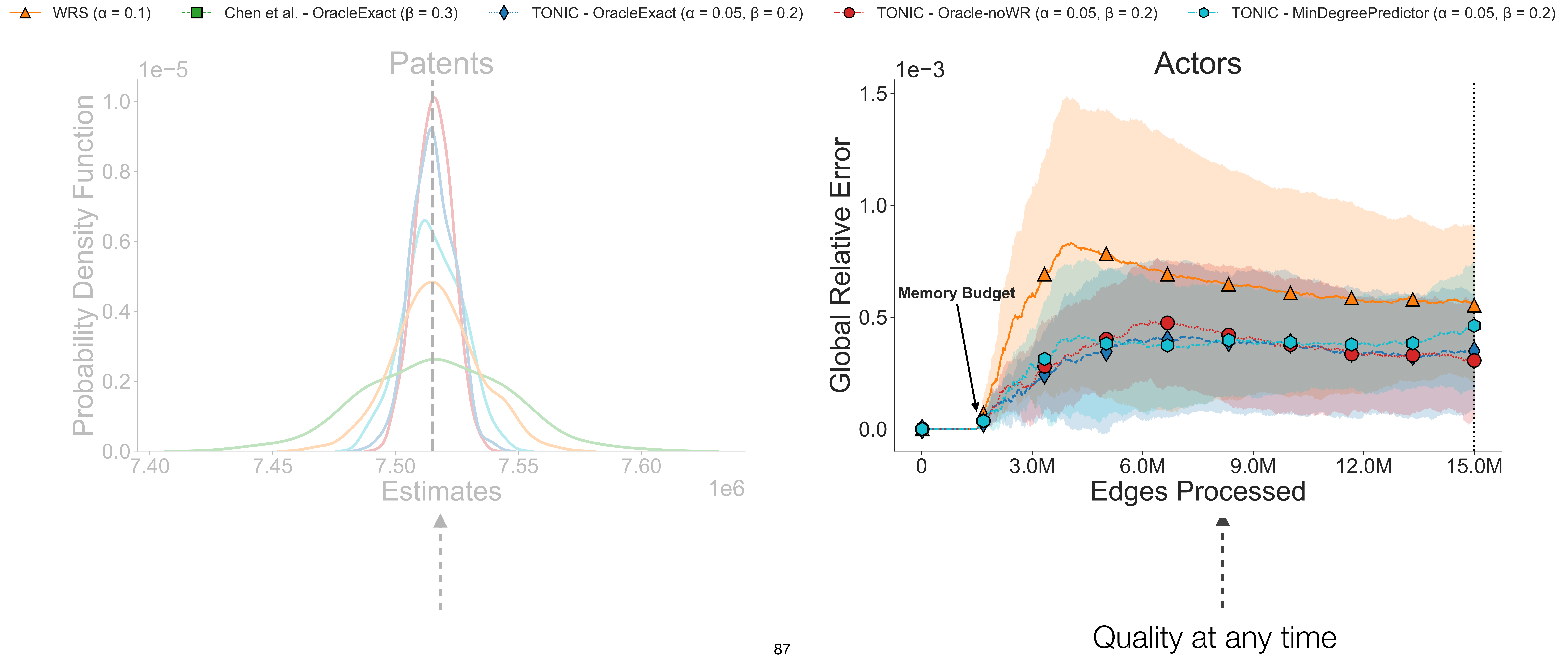
(1) Quality of approximations in terms of **unbiasedness** and **variance**



Smaller variance

Experimental Evaluation

(1) Quality of approximations in terms of **unbiasedness** and **variance**, and **estimations** at **any time** of the stream:



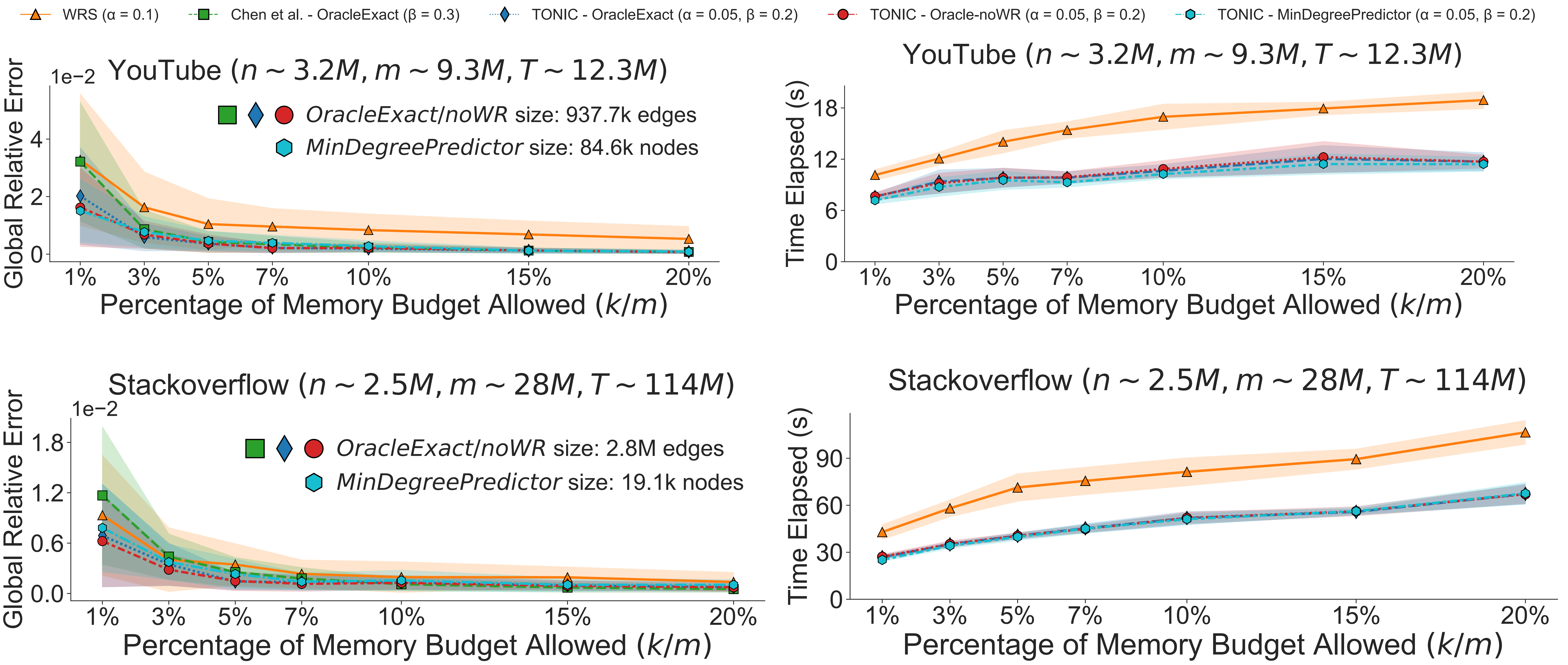
Experimental Evaluation

(2) Global Relative Error and Runtime vs Memory Budget k:

WRS (α = 0.1)   Chen et al. - OracleExact (β = 0.3)   TONIC - OracleExact (α = 0.05, β = 0.2)   TONIC - Oracle-noWR (α = 0.05, β = 0.2)   TONIC - MinDegreePredictor (α = 0.05, β = 0.2)

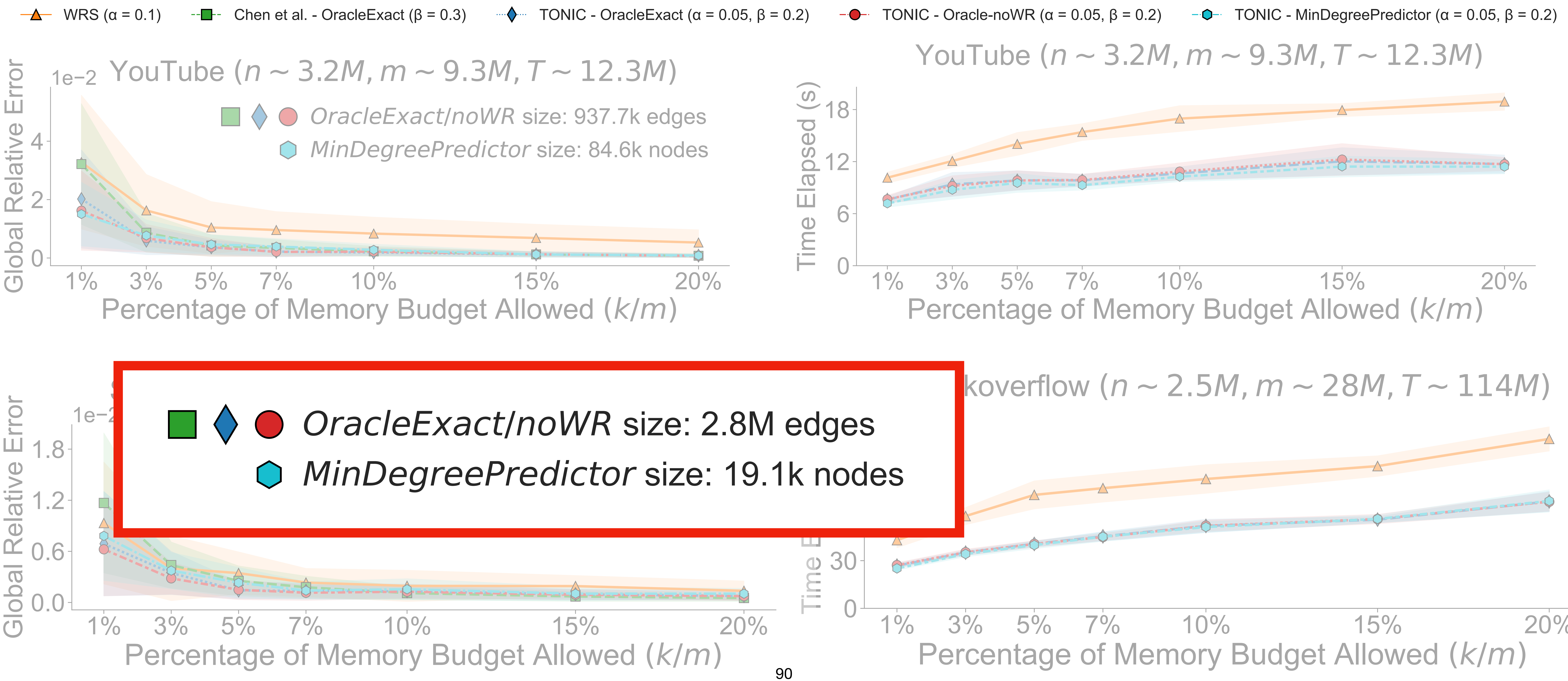
Experimental Evaluation

(2) Global Relative Error and Runtime vs Memory Budget k:



Experimental Evaluation

(2) Global Relative Error and Runtime vs Memory Budget k:



Experimental Evaluation

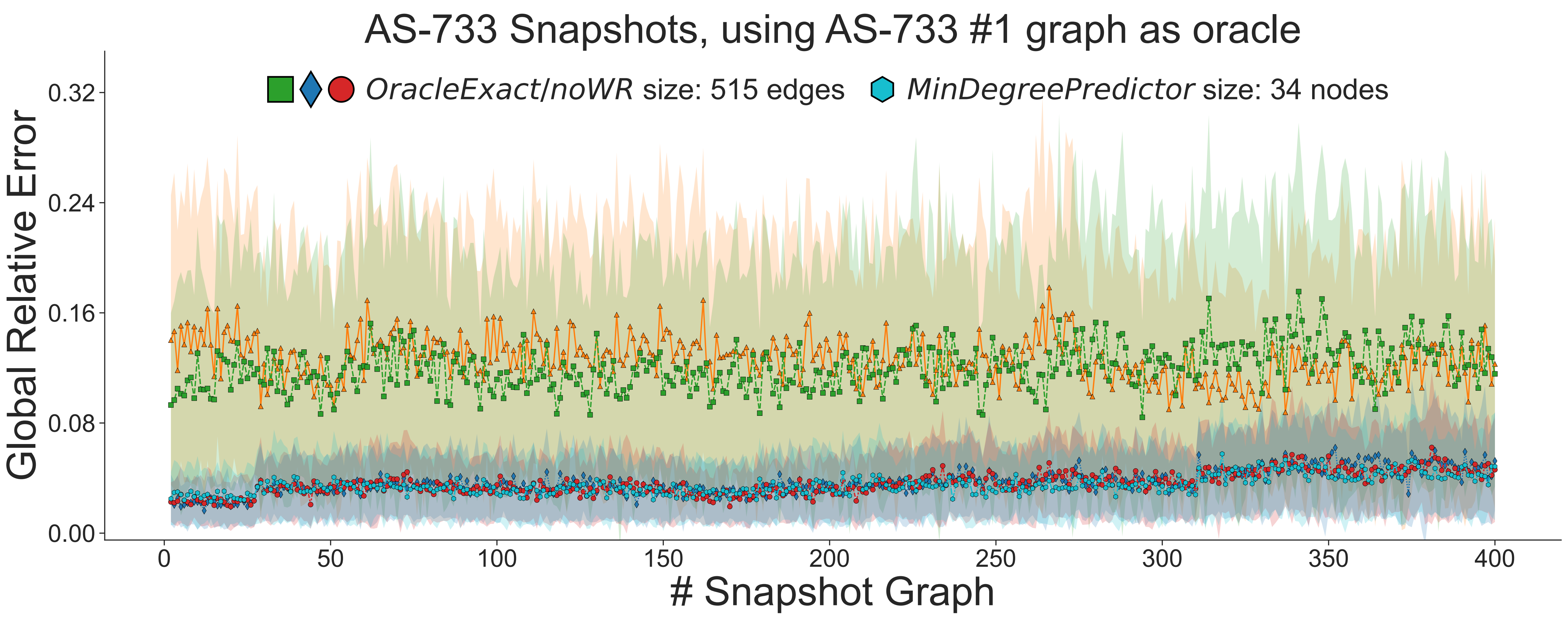
We consider **snapshot sequences** from autonomous system networks.

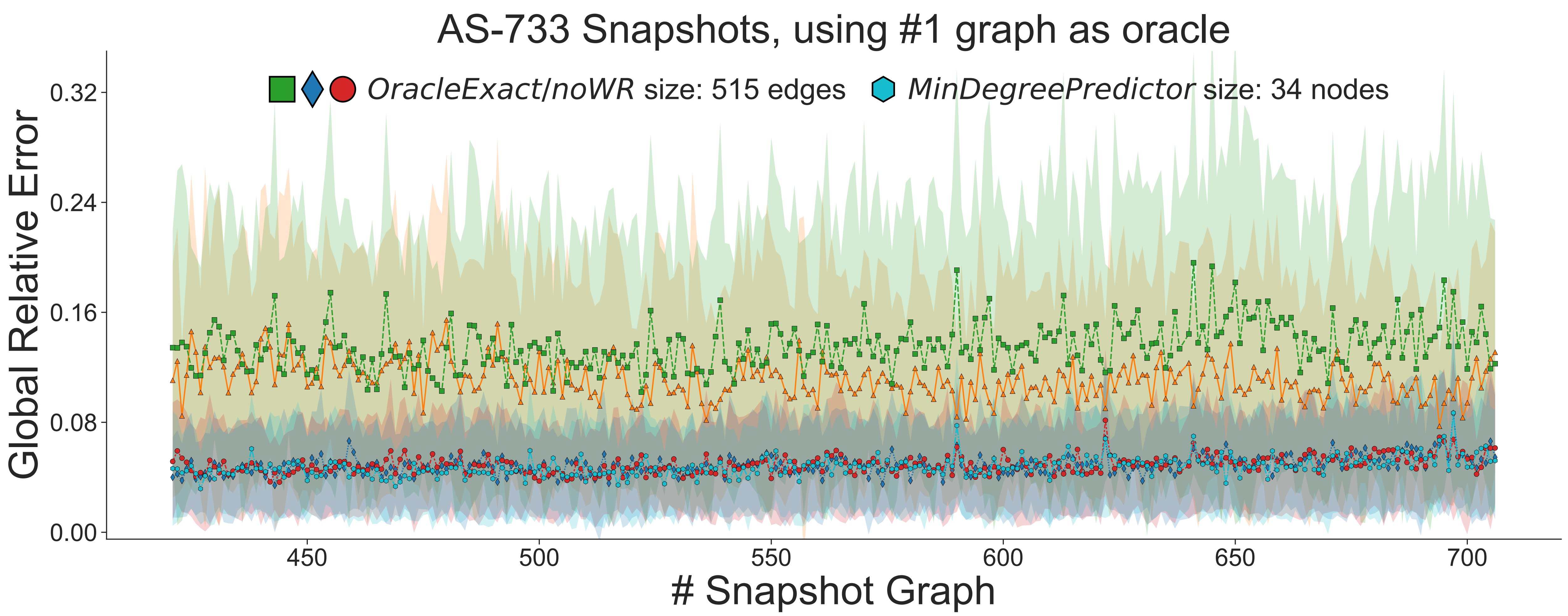
Snapshot Sequences			
Oregon (9 graphs)	$11k$	$23k$	$19.8k$
AS-CAIDA (122 graphs)	$26k$	$53k$	$36.3k$
AS-733 (733 graphs)	$6k$	$13k$	$6.5k$
Twitter (4 graphs)	$29.9M$	$373M$	$4.4B$

Predictors are trained only on the **first graph**, and then used for subsequent streams.

Experimental Evaluation

(3) Evaluation in **snapshot networks**:





Experimental Evaluation

(3) Evaluation in **snapshot networks**:

First ~ 400 streams

Experimental Evaluation

(3) Evaluation in **snapshot networks**:

Final ~ 300 streams

Conclusion

Our contributions:

- Fast and accurate algorithm for approximating the number of global and local triangles using predictions, both for insertion-only and fully-dynamic streams;

- Proposal of very simple and application-independent predictor, based on the degree of nodes;

- Extensive experimental evaluation, showing significant improvements, especially on networks with sequences of hundreds of graph streams.

Conclusion

Our contributions:

- Fast and accurate algorithm for approximating the number of global and local triangles using predictions, both for insertion-only and fully-dynamic streams;
- Proposal of very simple and application-independent predictor, based on the degree of nodes;
- Extensive experimental evaluation, showing significant improvements, especially on networks with sequences of hundreds of graph streams.

Thanks:

Progetto "National Centre for HPC, Big Data and Quantum Computing", CN00000013 (approvato nell'ambito del Bando M42C – Investimento 1.4 – Avvisto "Centri Nazionali" – D.D. n. 3138 del 16.12.2021, ammesso a finanziamento con Decreto del MUR n. 1031 del 17.06.2022)

UNIVERSITAS STUDII PADUANI MCCXXII

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

C. Boldrin and F. Vandin, **"Fast and Accurate Triangle Counting in Graph Streams Using Predictions"**, ICDM 2024.

Code and extended version of the paper:

arXiv https://arxiv.org/pdf/2409.15205

GitHub https://github.com/VandinLab/Tonic

cristian.boldrin.2@phd.unipd.it

Reservoir Sampling

Uniform sampling of edges in the stream [De Stefani et al., KDD 2016].

A sample $S \subseteq E$ is said to be an **uniform sample** if all equal-sized subsets of E are equally likely to be S

$$\mathbb{P}\left[S = A\right] = \mathbb{P}\left[S = B\right], \forall\, A \neq B \subseteq E \text{ such that } \left|A\right| = \left|B\right|.$$

Reservoir Sampling

Uniform sampling of edges in the stream [De Stefani et al., KDD 2016].

A sample $S \subseteq E$ is said to be an **uniform sample** if all equal-sized subsets of E are equally likely to be S

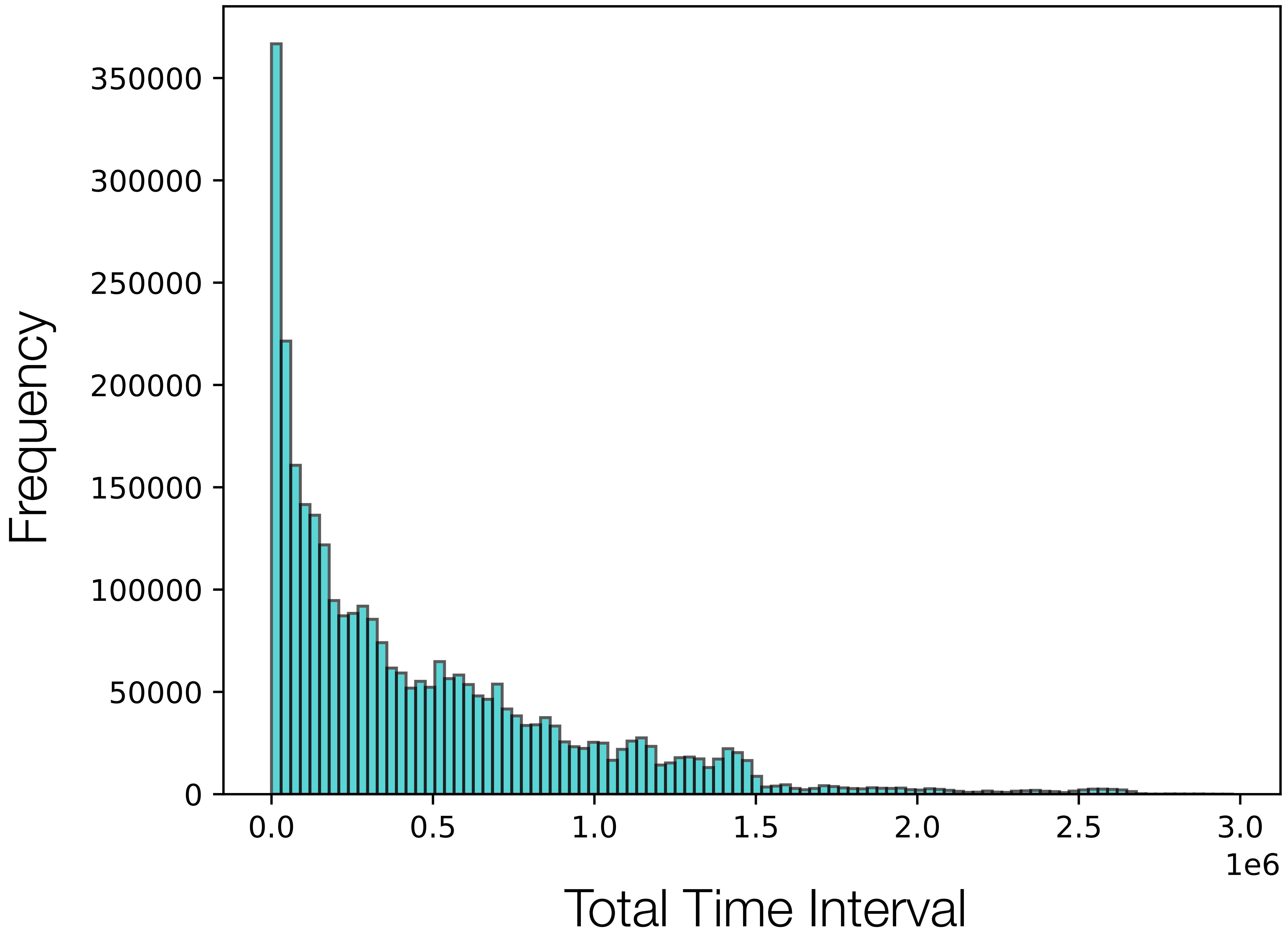
$$\mathbb{P}\left[S = A\right] = \mathbb{P}\left[S = B\right], \forall\, A \neq B \subseteq E \text{ such that } \left|A\right| = \left|B\right|.$$

Let $e^{(t)}$ be the edge at time t. **Reservoir sampling** keeps a **uniform sample** S of k edges as follows:

- If $\left|S\right| < k$, then $e^{(t)}$ is added to sample S
- Otherwise, with probability $\frac{k}{t}$, edge $e^{(t)}$ is added to sample S by replacing an uniformly at random edge from the sample

Waiting Room

Most real graph streams observe the tendency that future edges are more likely to form triangles with recent edges rather than with older edges [Shin K., ICDM 2017].

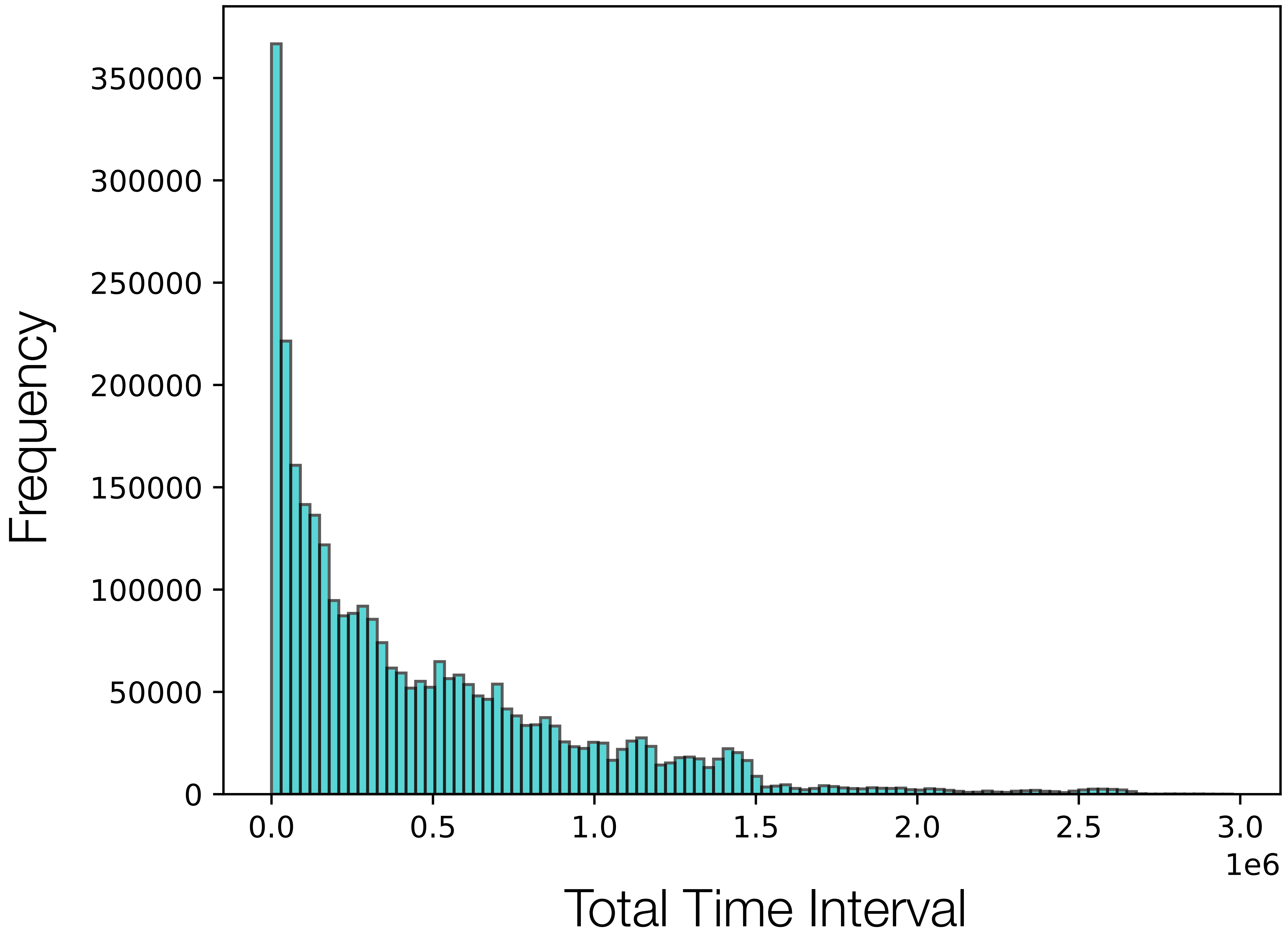


YouTube dataset

Total time interval: time between arrivals of first and last edge, for each triangle.

Waiting Room

Most real graph streams observe the tendency that future edges are more likely to form triangles with recent edges rather than with older edges [Shin K., ICDM 2017].



YouTube dataset

Total time interval: time between arrivals of first and last edge, for each triangle.

Always store the most recent edges in the waiting room W.

State of The Art

For **fully-dynamic** streams, we consider*:*

- ThinkD$_{acc}$ **:** *random pairing [Shin et al., ECML PKDD 2018]*
- WRS$_{del}$ **:** *waiting room + random pairing sampling [Lee et al., The VLDB Journal 2020]*

Random Pairing

Random Pairing: achieve **uniform sample** in fully-dynamic streams.

Goal: **compensate** sample **deletions** using subsequent insertions. Maintain counters d_g and d_b for number of good and number of bad *uncompensated* deletions.

When receiving an **edge insertion** $e^{(t)}$:

- If $d_g + d_b = 0$ (deletions compensated), then proceed by reservoir sampling
- Else, add $e^{(t)}$ to sample S with probability $\dfrac{d_b}{d_g + d_b}$ and decrement counters

When receiving an **edge deletion** $e^{(t)}$:

- If $e^{(t)}$ is not in the sample S, then ignore it (**good** sample **deletion**)
- Else, delete $e^{(t)}$ from S (**bad** sample **deletion**)

Algorithms with Predictions

Use of predictions about the input data has been formalised in the **“Algorithms with Predictions”** framework [Mitzenmacher and Vassilvitskii, 2020]

- Go beyond worst-case analysis
- Predictor empowering effectiveness of classical algorithms

Challenges:

- **Consistency:** useful predictions improve performances
- **Robustness:** bad predictions do not worsen too much performances
- **Practicality:** predictions derived by tasks on same data-domain

Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t.

Each triangle counted or deleted in the sample is scaled by the inverse of the probability with which the triangle edges but $e^{(t)}$ have been previously sampled.

Probability p Computation:

- If **no** edges are light: $p = 1$
- If only **one** edge is light: $p = p^{(t)}$
- If **both** edges are light: $p = p^{(t)} \cdot p^{(t-1)}$

Reservoir Sampling:

$$p^{(t)} = \min\left(1, \frac{k(1-\alpha)(1-\beta)}{\left|\mathcal{L}^{(t)}\right|}\right)$$

Tonic: Overall Algorithm

For each edge $e^{(t)}$ observed on the stream Σ at time t.

Each triangle counted or deleted in the sample is scaled by the inverse of the probability with which the triangle edges but $e^{(t)}$ have been previously sampled.

Probability p Computation:

- If **no** edges are light: $p = 1$
- If only **one** edge is light: $p = p^{(t)}$
- If **both** edges are light: $p = p^{(t)} \cdot p^{(t-1)}$

Random Pairing:

$$p^{(t)} = \min\left(1, \frac{k(1-\alpha)(1-\beta)}{\left|\mathscr{L}^{(t)}\right| + d_g + d_b}\right)$$

Tonic: theoretical analysis

We prove that the predictor helps when it provides fairly reliable information on heavy edges.

Tonic: theoretical analysis

We prove that the predictor helps when it provides fairly reliable information on heavy edges.

If this is not the case, we also prove that our algorithm *Tonic* returns estimates as accurate as *WRS*.

Proposition (informal): the variance of the estimates of *Tonic* is equal than the variance of the estimates of *WRS* if the predictor predicts a randomly chosen set of edges as heavy edges.

Tonic: theoretical analysis

Let T_H the total number of triangles in which **heavy** edges appear, and T_L similarly for **light** edges.

Proposition (informal): the variance of *Tonic* estimates is less than the variance of *WRS* estimates if:

$$T_H > 3 \; \frac{(1/p'^2 - 1/p^2) + c\rho(1/p' - 1/p)}{(1/p - 1)(3 + 4\rho/c)} \cdot T_L$$

Interpretation: *useful* predictions (predicted *heavy* edges are involved in a sufficient number of triangles), lead to better estimates.

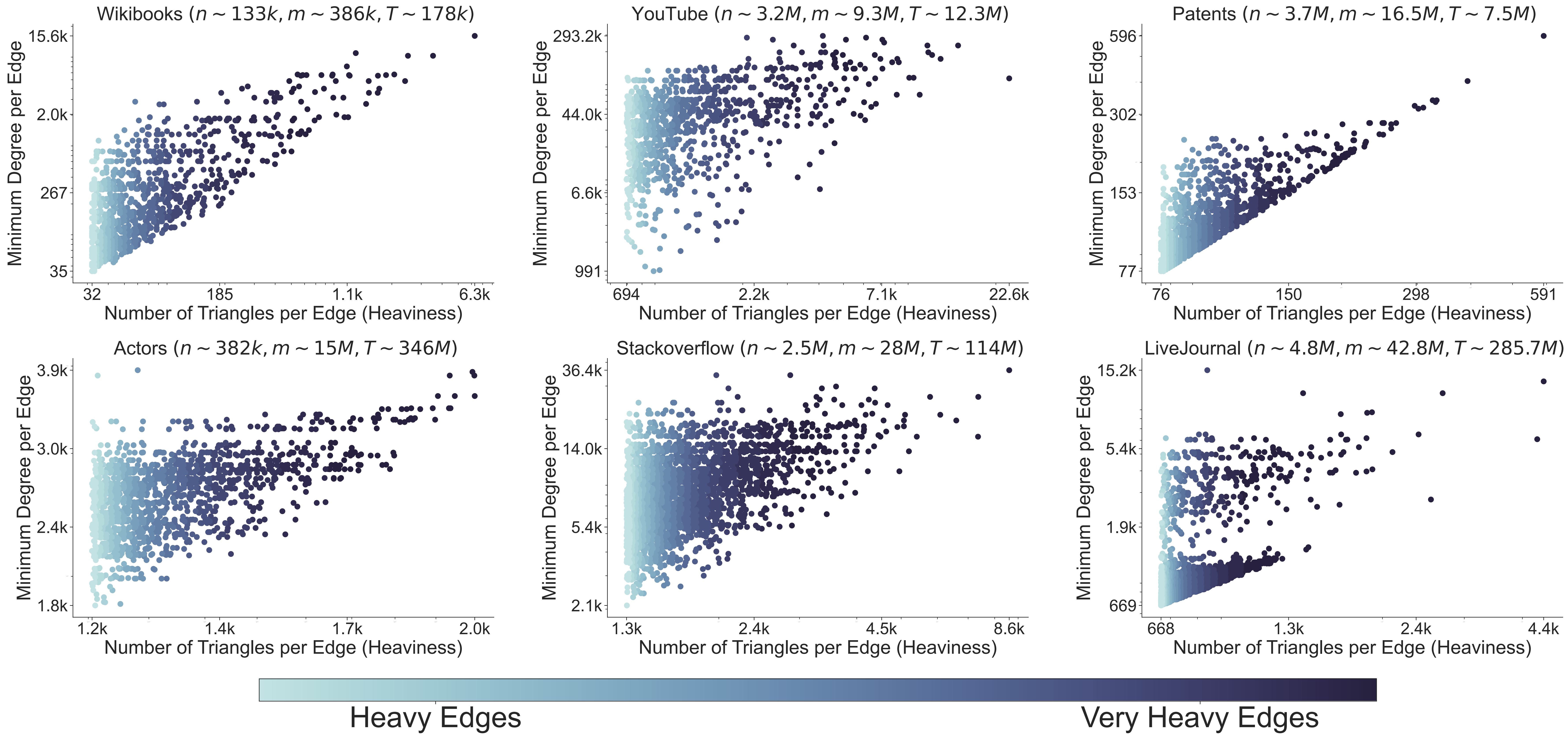
Representative values: if $p' = 0.09 < p = 0.1$, $\rho = 100$ and $c = 1.5$, then the bound above corresponds to T_H being at least one fifth of the overall number of triangles.

Edge Heaviness Predictor

The predictor used by *Tonic* could be implemented by a machine learning model that may consider information other than the graph.

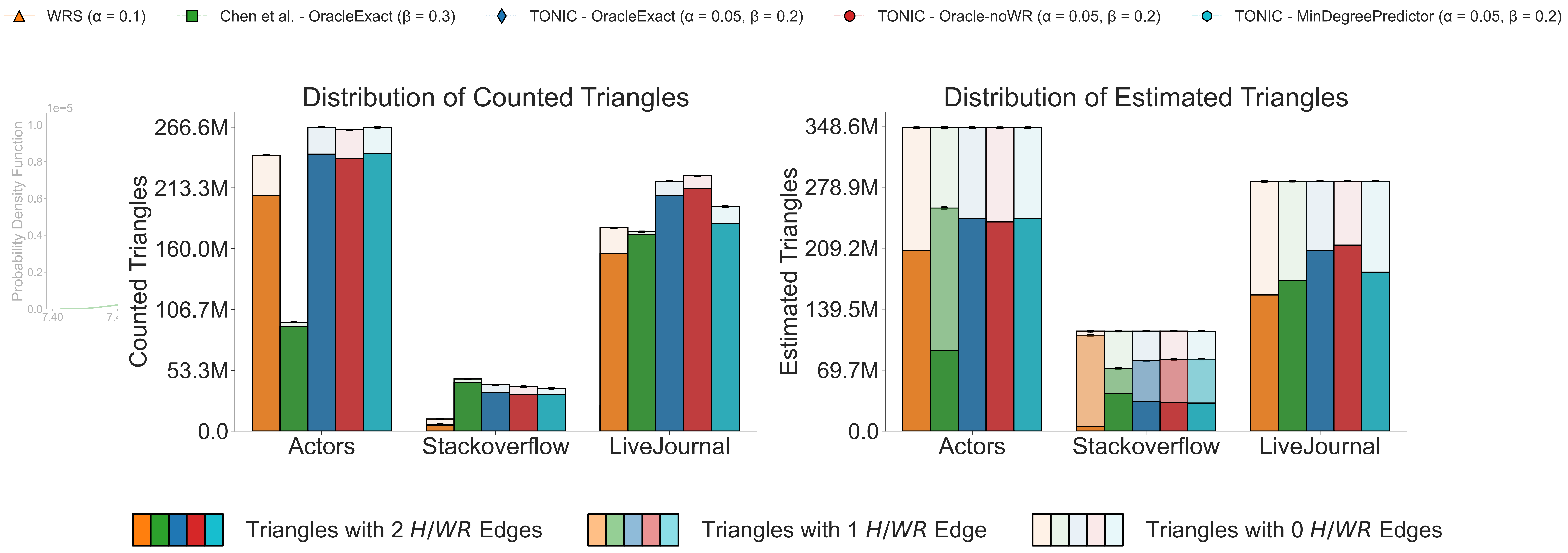
In our experiments we consider:

- *OracleExact*
- *Oracle-noWR*
- *MinDegreePredictor*



Experimental Evaluation

(*) Quality of approximations in terms of unbiasedness and variance, estimations at any time of the stream, and **number** of **counted** and **estimated** triangles.



Experimental Evaluation

(4) Performances in **fully-dynamic** streams.

Streams are created computing additions and deletions from snapshot networks.

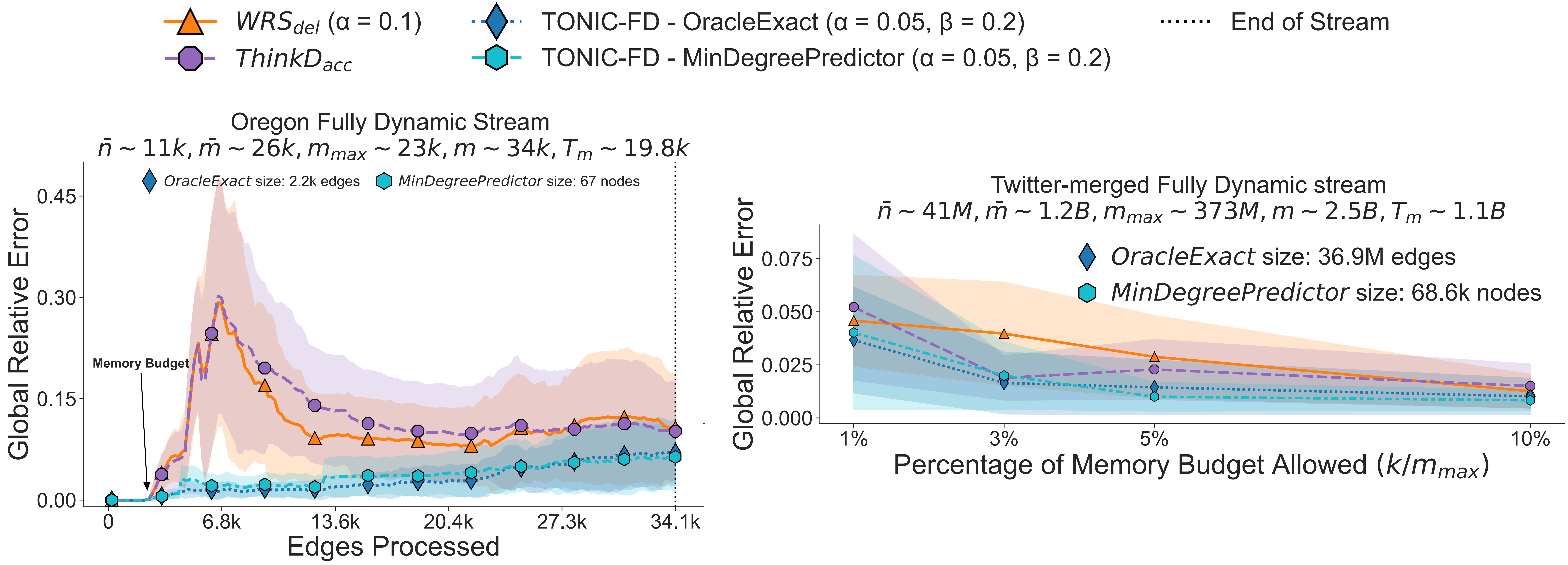
Again, predictors are trained only on the first graph, hence oblivious to removals of edges.

Experimental Evaluation

(4) Performances in **fully-dynamic** streams.

Streams are created computing additions and deletions from snapshot networks.

Again, predictors are trained only on the first graph, hence oblivious to removals of edges.



Quality at any time

Better estimates